前 言

S3C9444/F9444/C9454/F9454 用户手册是专门为开发，SC9444/F9444，S3C9454/F9454 微控制器的底层设计人员或编程人员而编写的一部参考书,它主要包括两部分：

第一部分: 软件特性 第二部分： 硬件特性

第一部分，主要介绍 S3C9444/F9444/C9454/F9454 控制器的结构、编程形式、指令集、和中断结构，让使用者熟悉与底层硬件编程有关的知识。它包括 6 个章节：

第一章：	概述	第四章：	控制寄存器
第二章：	地址空间	第五章：	中断
第三章：	地址空间访问模式	第六章：	SAM88RCRI 指令集

第一章，概括的介绍了 S3C9444/F9444/C9454/F9454 的硬件资源，详细说明了各引脚特性和引脚的电路类型。

第二章，介绍了 S3C9444/F9444/C9454/F9454 的程序存贮器，数据存贮器，内部寄存器卷和控制寄存器，以及怎样访问它们；同时，也介绍了通用寄存器工作组和堆栈的操作。

第三章，详细地介绍了 S3C9444/F9444/C9454/F9454 CPU 内核所支持的六种地址访问模式。

第四章，介绍了 S3C9444/F9444/C9454/F9454 的控制寄存器，并用标准形式的表格详细的列写了内部寄存器和外围引脚控制寄存器。方便参阅者按照字母顺序查阅各控制寄存器和外围引脚的功能控制。

第五章，介绍了 S3C9444/F9444/C9454/F9454 的中断结构及其硬件电路。

第六章，介绍了适用于 S3C9-系列的每条指令的功能及其应用。同时提供了一些表格以供参考。在这一章，每条指令都进行了详细的讲解，并给出了相应的例程。

第二部分，详细地介绍了 S3C9444/F9444/C9454/F9454 微控制器的外围组成元件，同时还给出了机械、电气、MTP、及开发工具等必需的数据资料。它包括 10 个章节：

第七章：	时钟电路	第十二章：	A/D 转换
第八章：	复位和掉电保护	第十三章：	电气特性
第九章：	输入输出口	第十四章：	机械尺寸
第十章：	实时时钟和定时器	第十五章：	S3F9444/F9454
第十一章：	脉宽调制		

目 录

第一章 概述

三星 SAM88RCRI 是一款单片 8 位 CMOS 型微控制器，她向用户提供了高效快速的 CPU 处理，丰富的外围接口，以及各种类型的可编程 ROM。她的数据/地址总线结构和位编程 I/O 口提供了一个灵活的编程环境，能够满足不同用户对存贮器和 I/O 口的要求。同时，还可以实时选择定时/计数器的工作模式。

S3C9444/C9454 微控制器

S3C9444/C9454 是一款 8 位微控制器，适用于 A/D 转换，SIO 等应用领域。她采用了 SAM88RCRI 内核和自己独有的内部结构，扩展了片内寄存器卷，增加了内部数据寄存器空间。S3C9444/C9454 有 2K/4K 字节的片内 ROM 和 208 字节的 RAM。

S3C9444/C9454 是一款多功能，通用型微控制器，她是定时/计数、PWM、多路 A/D 转换等应用领域的理想选择。另外，S3C9444/C9454 内部特有的 CMOS 技术使芯片呈现出低功率消耗和宽电压工作范围的特性。

S3C9444/C9454 在 SAM88RCRI 内核上集成了以下外围接口：

-- 3 个可编程 I/O 口（18 个引脚）

-- 4 个中断源，同一中断优先级，一个中断向量

-- 1 个 8 位定时/计数器

-- 9 路模数转换器，10 位转换结果

-- 1 个 8 位 PWM 输出

S3F9444/F9454 是一款可多次编程的微控制器,她片内有 4K 可多次擦写的 FLASH ROM ，S3F9444/F9454 在直流电气特性和引脚特性上与 S3C9444/C9454 是完全相同的。

S3C9444/C9454 微控制器特性

CPU
- SAM88RCRI 内核

存贮器
- 2K/4K 内部程序存贮器
- 208 个字节的通用数据寄存器

指令集
- 41 条指令

指令执行周期
- 在晶振为 10M 情况下，最小为 400ns

中断
- 四个中断源，一个中断向量
- 同一中断优先级

I/O 口
- 3 个 I/O 口（最多 18 管脚）
- 各口引脚都可位编程

高速 PWM
- 一路 8 位高速 PWM(最大:156KHZ)
- 6 位数据比较,2 位 PWM 脉冲延伸控制位

内部复位电路
- 低电压检测复位电路

定时/计数
- 一个可用作看门狗的 8 位 Basic Timer
- 一个定时/计数器

A/D 转换
- 9 路 A/D 输入
- 10 位 A/D 转换结果

晶振频率
- 1M～10M 外部晶振
- 最大 10M 的 CPU 时钟输入
- 在 5V 工作电压下,内部 RC 震荡 3.2M，0.5M 的时钟频率

温度工作范围
- -40℃～+85℃

电压工作范围
- 2.0V～5.5V

可得到的封装形式
- S3C9444B/C9454B：
  20-DIP-300A
  20-SSOP-225
  20-SOP-375
  16-SOP-BD300-SG
  16-DIP-300A
  16-SOP
  16-SSOP-BD44

结构示意图

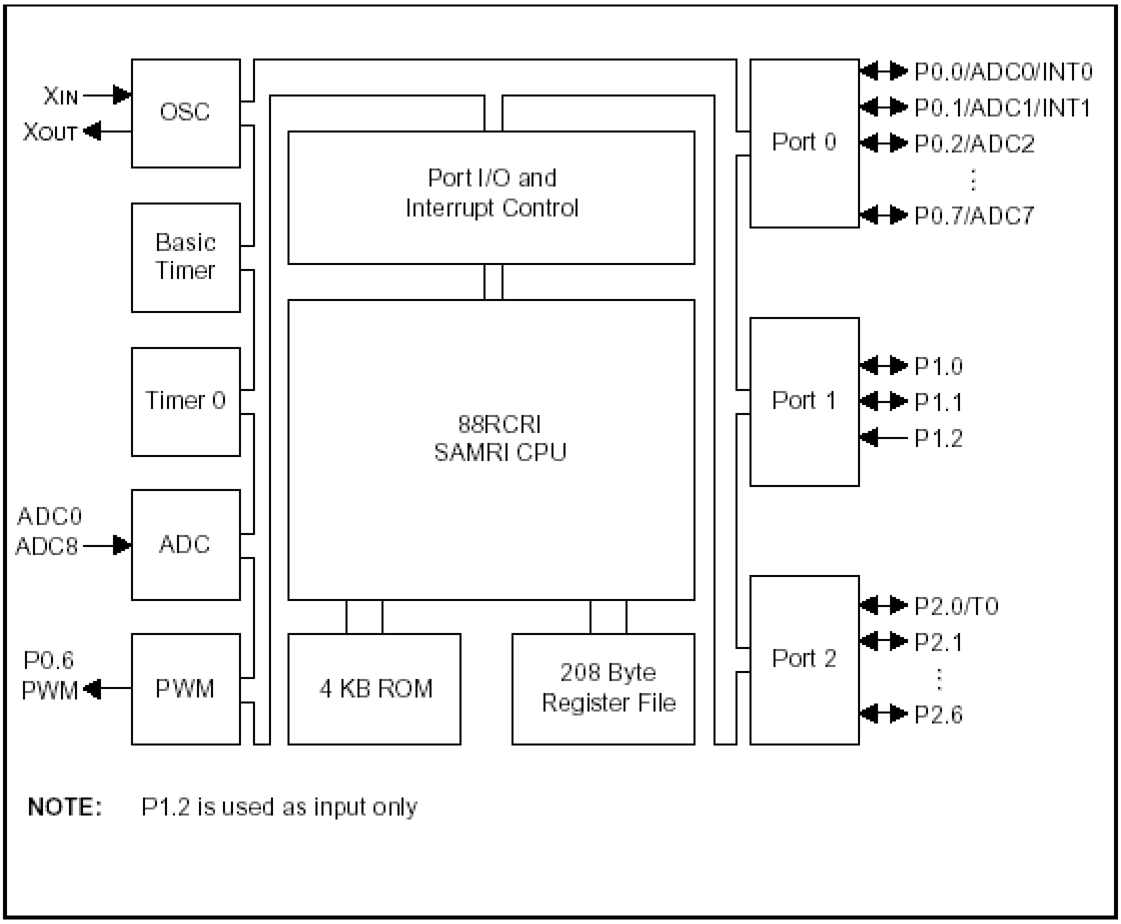


图 1-1 原理图

引脚分配

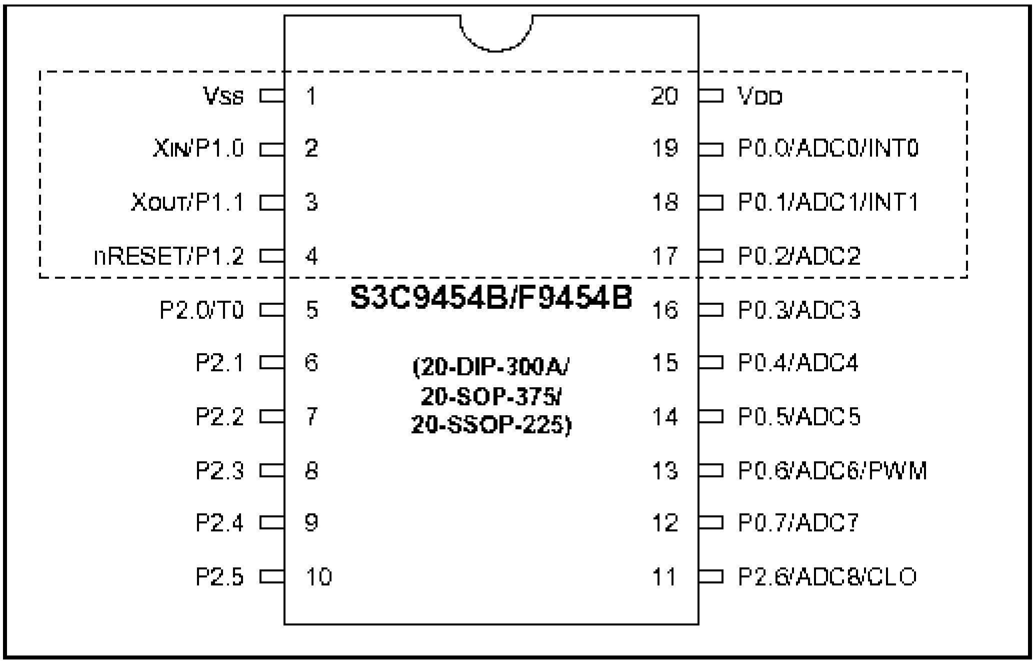


图 1-2 引脚分布图（20-DIP/SOP/SSOP 封装）

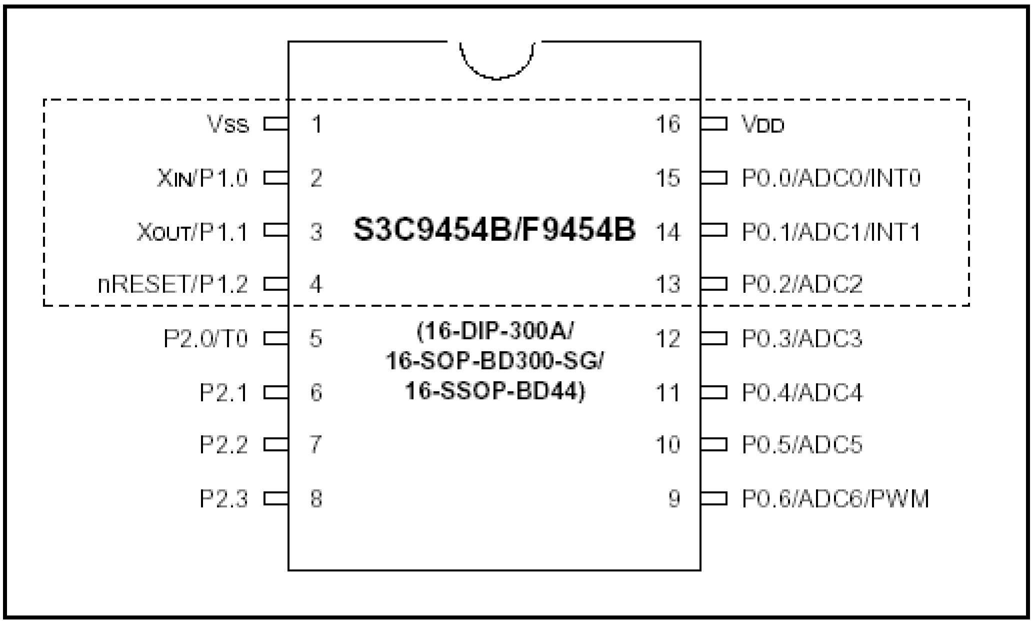


图 1-3 引脚分布图（16-DIP/SOP/SSOP 封装）

引脚特性描述

表 1　 S3C9454 引脚特性

引　脚 名　称	输入/ 输出	引 脚 特 性 描 述	引　脚 类　型	共 用 引 脚
P0.0-P0.7	I/O	可位编程该 I/O 口为施密特触发器输入或推拉式输出。上拉电阻可用软件设定。P0 口也可用作 A/D 转换输入，PWM 输出或外部中断输入	E--1	ADC0-ADC7 INT0/INT1 PWM
P1.0-P1.1	I/O	可位编程该 I/O 口为施密特触发器输入或推拉式输出，开漏输出。上拉电阻或下拉电阻均可用软件设定。	E--2	X_{IN} ,X_{OUT}
P1.2	I	施密特触发器输入	B	RESET
P2.0-P2.6	I/O	可位编程该 I/O 口为施密特触发器输入或推拉式输出，开漏输出。上拉电阻可用软件设定。	E E--1	- ADC8/CLO T0
X_{IN} ,X_{OUT}	-	石英晶振/陶瓷晶振，或 RC 振荡作为系统时钟信号节拍		P1.0-P1.1
nRESET	I	内部 LVR 或外部复位电路	B	P1.2
V_{DD},V_{SS}	-	电源和地输入引脚		-
CLO	O	系统时钟输出口	E--1	P2.6
INT0-INT1	I	外部中断输入引脚	E--1	P0.0-P0.1
PWM	O	8 位高速 PWM 输出口	E--1	P0.6
T0	O	T0 定时输出口	E--1	P2.0
ADC0-ADC8	I	A/D 转换输入口	E--1 E	P0.0-P0.7 P2.6

引脚电路

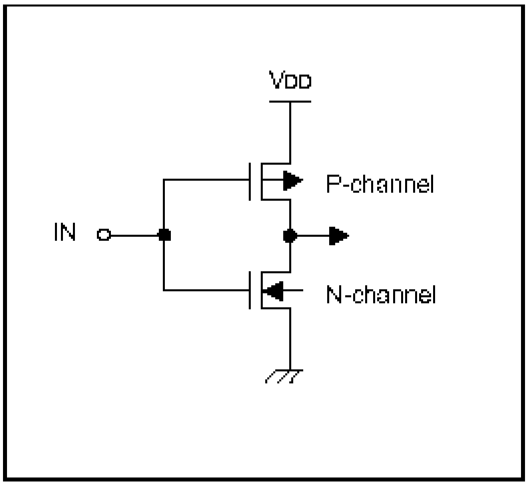


图 1-5 电路类型 A

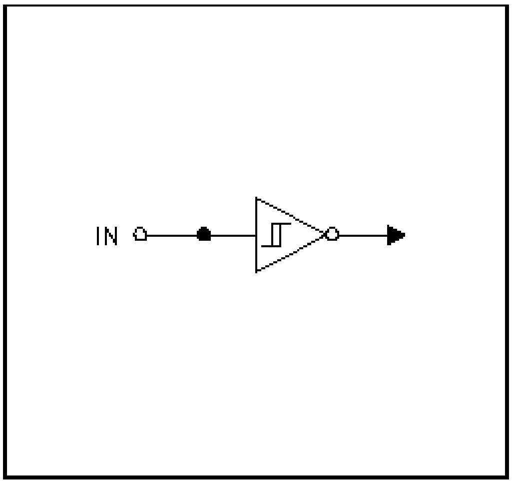


图 1-6 电路类型 B

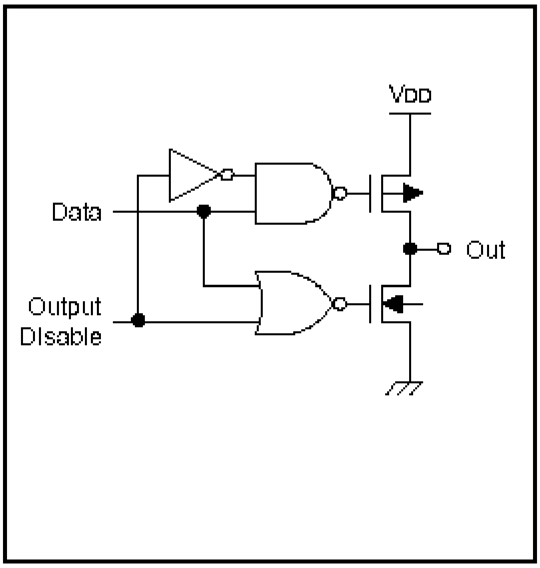


图 1-7 电路类型 C

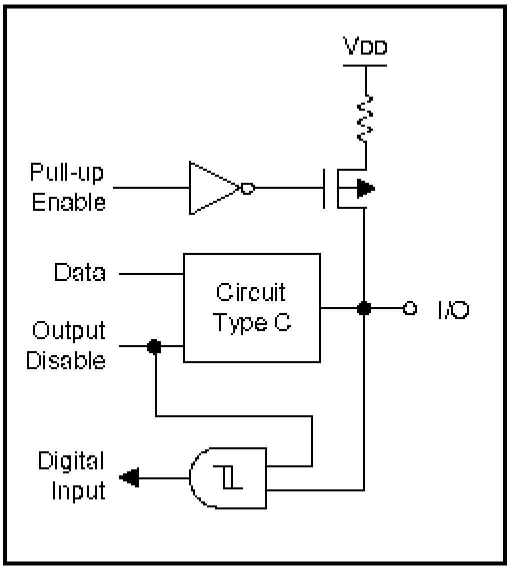


图 1-8 电路类型 D

（续）

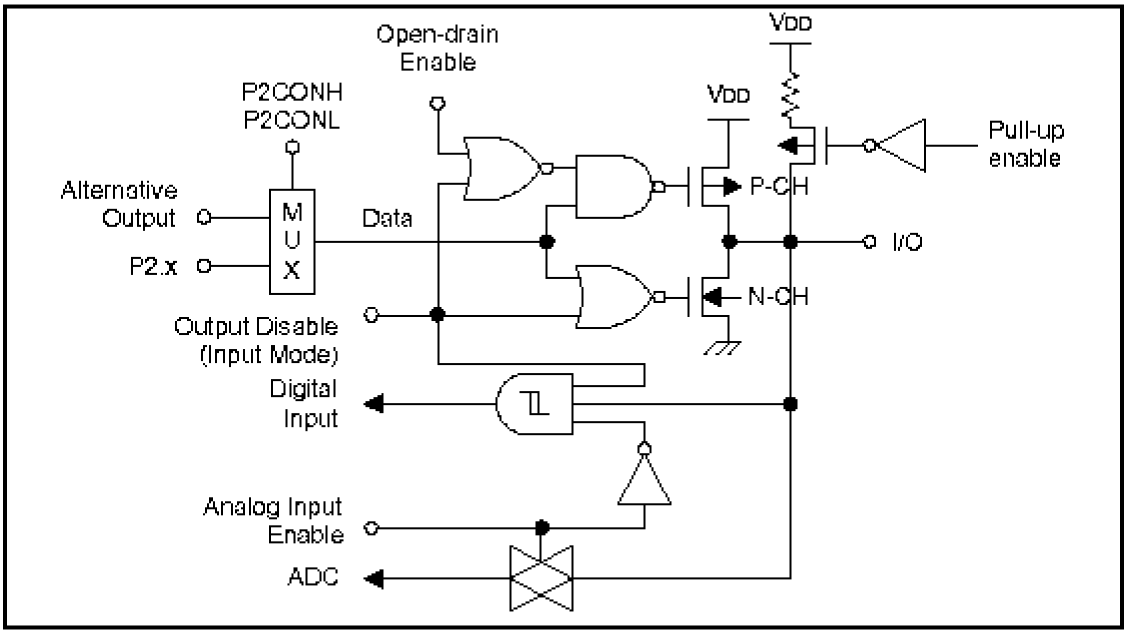


图 1-9 电路类型 E

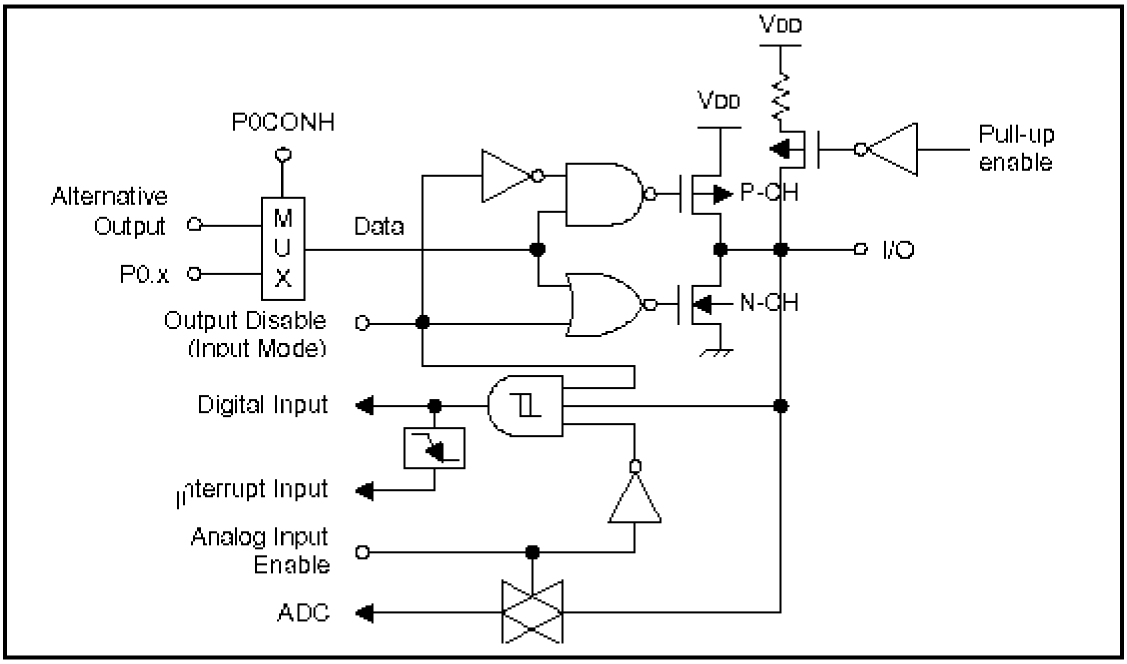


图 1-10 电路类型 E-1

(续)

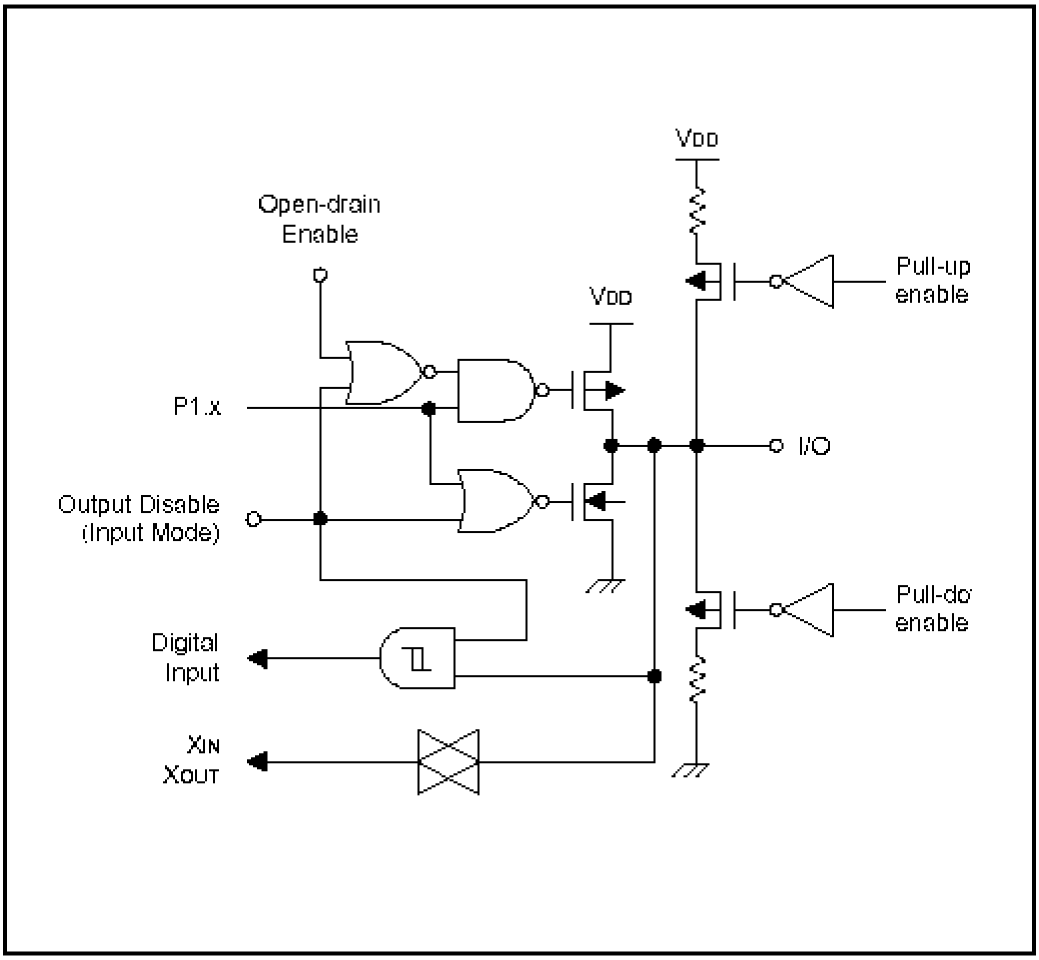


图 1-11　电路类型 E-2

第二章　　地址空间

概述

S3C9444/C9454 微控制器芯片有两类地址空间：

—— 内部程序存贮空间（ROM）

—— 内部寄存器卷

微控制器通过 12 位的地址线访问程序存贮空间，通过独立的 8 位地址线和数据线访问内部寄存器卷。S3C9444/C9454 有可完全编程的 2K/4K 片内掩模 ROM，有 208 个字节的通用寄存器，26 个系统控制寄存器和外围接口控制寄存器。

程序存贮空间（ROM）

S3C9444/C9454 有 2K（0H--07FFH）/4K（0H--0FFFH）的内部掩模（ROM）。起始 2 个字节（0000H—0001H）是中断入口地址。除去 3CH、3DH、3EH、3FH 等存贮空间，从 0002H—00FFH 可以用作普通的存贮空间。3CH、3DH、3EH、3FH 是 SMART OPTION，用作外围端口及片内状态初始化选择。

控制器复位后，系统从 0100H 开始执行程序。

存贮空间示意图如下：

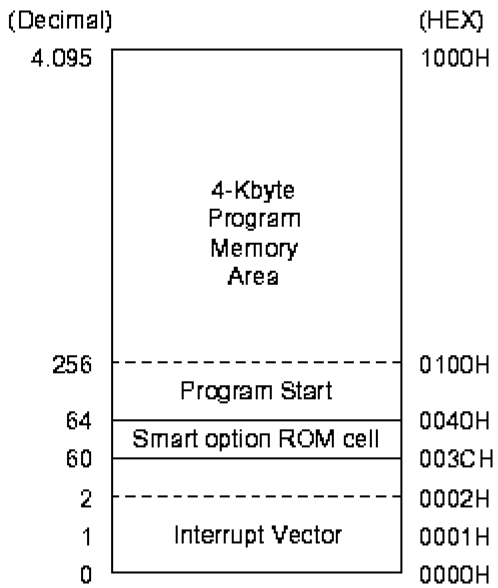


图 2-1　程序地址存贮空间

ROM 中的 SMART OPTION 用于初始化片内状态和端口工作状态。

ROM 中用于 SMART OPTION 的地址范围为 003CH--003FH，共有 4 个存贮单元。S3C9442/C9444/C9452/C9454 只用了 003EH 和 003FH，没有用 003CH、003DH，没**有用到的存贮空间应该初始化为“00H”，默认情况下，ROM 存贮空间初始值为“FFH”（允许 LVR 和内部 RC 震荡时钟）。**

SMART OPTION 空间的内容如下：

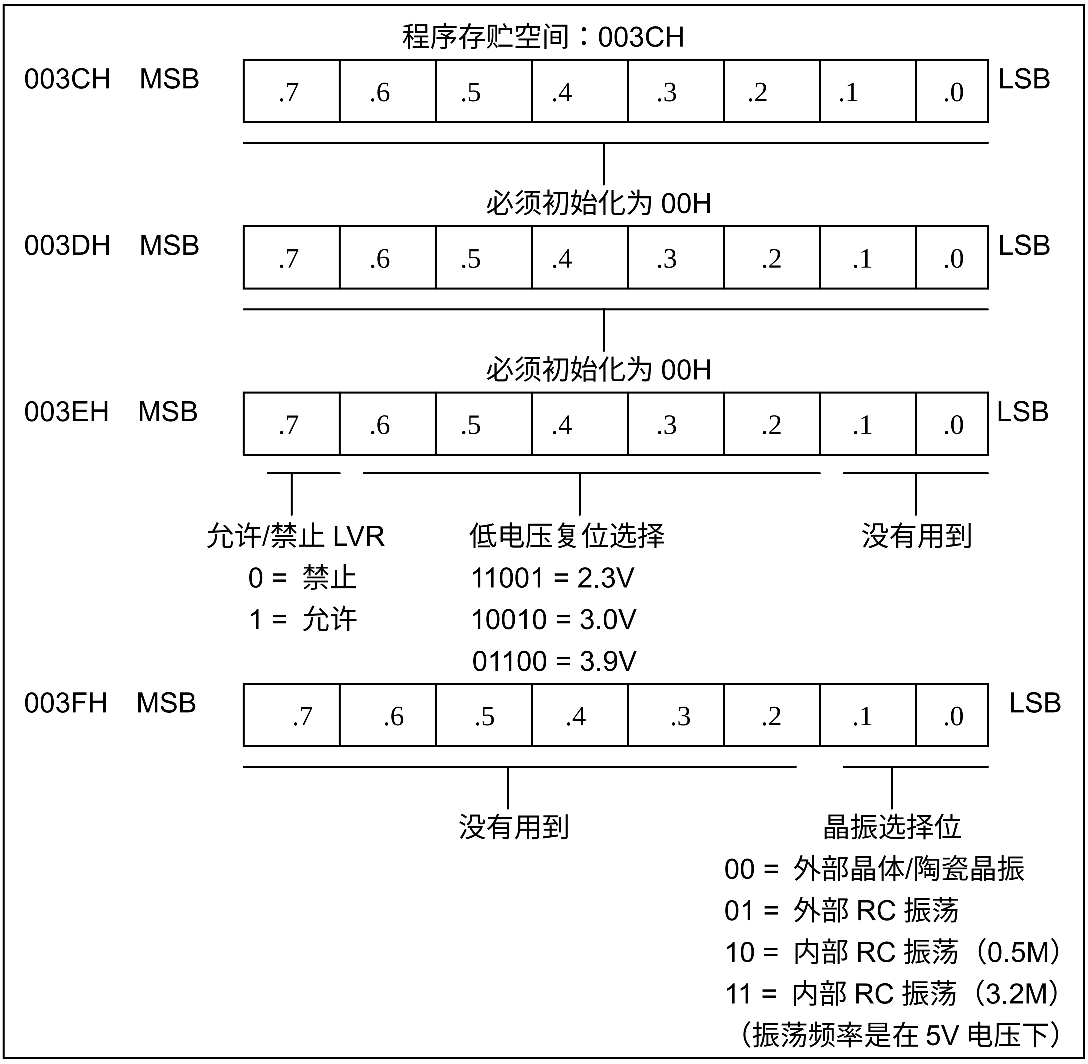


图 2-2 smart option

当使用外部晶振时，为减小电流消耗，P1.0、P1.1 引脚必须设置为输出模式。

可以不用关心 3EH,3FH 中没有用到的位，即没有用到的位可以设置为任意值。

当允许低电压（LVR）复位时，低电压(LVR)值应该设置为相应的值，不应该为缺省值。

☞ 程序例程：

```
;           中断向量地址
            ORG         0000H
            VECTOR      00H，INT_9454           ；中断服务程序入口地址
            ORG         003CH
            DB          00H                     ；003CH 必须初始化为 0
            DB          00H                     ；003EH 必须初始化为 0
            DB          0E7H                    ；允许低电压复位 LVR（2.3V）
            DB          03H                     ；内部 RC 晶振（3.2M，VDD = 5.0V）
            .
            .
            .
;           <>
            ORG         0100H                   ；复位后，程序起始地址
RESET:      DI                                  ；复位初始化程序
            LD          SP,#0C0H
            .
            .
            .
            .
            .
INT_9454:PUSH           R0                      ；中断服务子程序
            PUSH        ..
            .
            .
            .
            POP         ..
            POP         R0
            IRET
```

寄存器结构

S3C9444/C9454 内部寄存器卷中的高 64 位用作工作寄存器组，系统控制寄存器和外围接口控制寄存器；从 00H-0BFH 的低 192 个字节用作通用寄存器，供用户使用。寄存器卷中共有 234 个寄存器可以访问；有 208 个寄存器可以供用户使用。

对于 SAM88RCRI 微控制器，在内部寄存器卷中，都已扩展了寄存器页以方便使用。但在 S3C9444/C9454 中没有扩展页，只有一个页（00h—BFH: page0）可以使用。

下表中总结了内部寄存器卷中特殊功能寄存器类型以及它们占用的字节以供参考。

寄存器类型总结

寄存器类型	所占字节
CPU，系统控制寄存器	11
外围接口，I/O 口，时钟控制器，数据寄存器	15
通用寄存器（包括 16 个寄存器工作组）	208
所占字节共	234

内部寄存器卷的地址安排

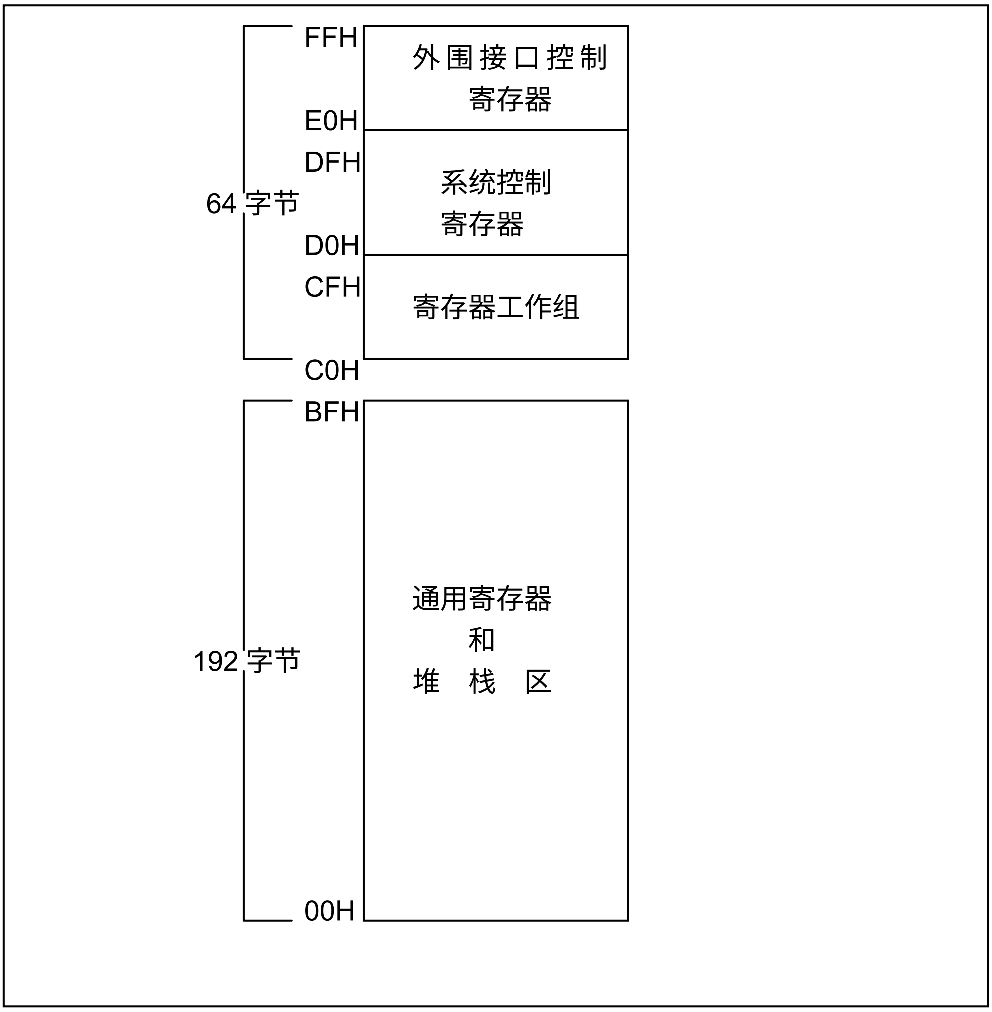


图 2-3　　内部寄存器卷的地址空间

工作寄存器

为充分利用短指令格式的优点，减少指令执行时间，SAM88RCRI 寄存器结构提供了一个有效的工作寄存器访问模式。

0C0H—0CFH 这 16 个地址范围称为通用寄存区，这个地址范围的寄存器可以用作工作寄存器，工作寄存器可以访问寄存器卷任意页中的任何地址空间。典型的，这些工作寄存器可以用作不同页之间数据操作的数据暂存器。但 S3C9444/C9454 只有第 0 页，可以用工作寄存器进行任意的内部数据操作。寄存器访问模式可以访问这些存贮区。

工作寄存器既可以用作 8 位寄存器也可以用作 16 位寄存器。**当工作寄存器用作 16 位时，开始地址必须为偶数，下一个地址则为奇数。**在读写数据时，高字节数据存在偶数地址寄存器，而低字节数据存在奇数地址寄存器。

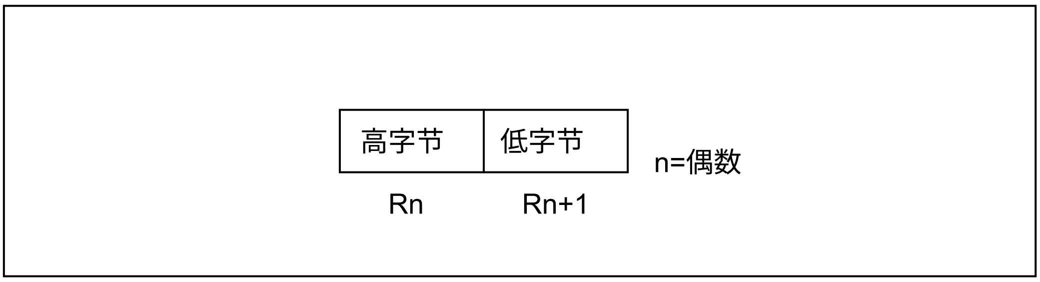


图 2-4　16 位寄存器结构

☞ 访问通用寄存器的编程实例：

下面的例子，运用寄存器访问模式来访问位于 C0H—CFH 地址的通用寄存器。

```
1、  LD    0C2H，40H      ；非法访问模式
     LD    R2，40H        ；R2←（40H），正确的访问方法
2、  ADD   0C3H，#45H     ；非法访问模式
     ADD   R3，#45H       ；R3←（R3）+45H
```

系统堆栈

S3C9-系列微控制器利用系统堆栈实现子程序调用、返回，及数据存放。S3C9444/C9454 内部结构支持内部寄存器卷的堆栈操作。PUSH，POP 指令来实现堆栈的操作。

中断返回地址，子程序调用返回地址和压栈数据存放在堆栈区。当有 CALL 指令时，PC 值被压入堆栈，当有 RET 指令时，PC 值弹出堆栈。当有中断时，PC 值和 FLAGS 值被压入堆栈，当有 IRET 指令时，把引入中断时的 PC 值和 FLAGS 值弹到原来的寄存器。**在压栈时，堆栈地址自动减一，而后压栈；在弹栈时，当数据弹出后，堆栈地址自动加一。**堆栈地址的指针始终指向栈顶的位置。

堆栈示意图：

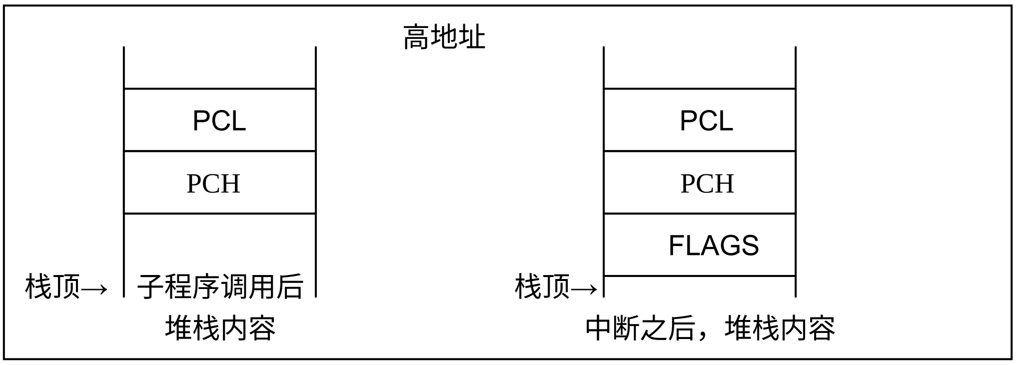


图 2-5　　堆栈操作

堆栈指针（SP）用于系统堆栈操作，地址为 D9H。系统复位后，堆栈指针的值不确定。在程序初始化中，SP 必须初始化为范围在 00H-0CH 中间的 8 位数据。

提示：**假如堆栈指针（SP）初始化为 00H，当堆栈开始操作时，SP 的值变为 0FFH，这意味着系统访问了非法的堆栈区。在程序初始化时，我们建议把堆栈指针初始化为 0C0H，这样堆栈操作时，系统访问地址为通用寄存器的高地址—0BFH**。

☞ 堆栈操作例程

下面的例子，告诉你怎样用 PUSH、POP 指令对寄存器卷中的堆栈区进行操作。

```
LD      SP，#0COH        ;SP←C0H，通常情况下，SP 初始化为 0C0H
.
.
.
PUSH    SYM              ;0BFH←SYM
PUSH    R15              ;0BEH←（R15）
PUSH    20H              ;0BDH←（20H）
PUSH    R3               ;0BCH←（R3）
.
.
.
POP     R3               ;（0BCH）→SYM
POP     20H              ;（0BDH）→20H
POP     R15              ;（0BEH）→R15
POP     SYM              ;（0BFH）→SYM
```

第三章　　　地址访问空间

概述

S3C9444/C9454 通过程序计数器来访问程序存贮空间的指令，指令隐含着要执行的操作和数据处理。地址访问模式是用于决定操作数地址的一种方法。SAM88RCRI 指令确定的操作数可能是条件转移指令，立即数或者是寄存器卷，程序存贮区，数据存贮区中的地址。

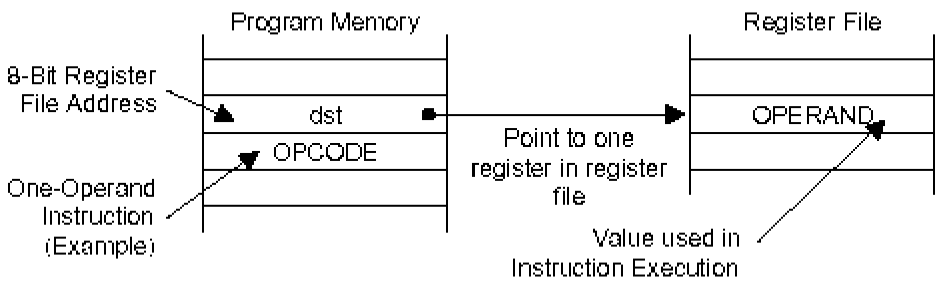
S3C9444/C9454 指令集支持六种地址访问模式，但并不是所有的指令都支持地址访问。下面给出具体的地址访问模式和它们的符号表示：

— 寄存器访问模式（R）
— 间接寄存器访问模式（IR）
— 偏址访问模式（X）
— 直接访问模式（DA）
— 相对地址访问模式（RA）
— 立即数访问模式（IM）

寄存器访问模式

在寄存器访问模式中，操作数是具体寄存器中的值（如图 3-1）。工作寄存器访问模式与寄存器访问模式是不同的，这是因为工作寄存器用寄存器卷中的一个 16 位寄存器和一个 4 位寄存器（如图 3-2）。

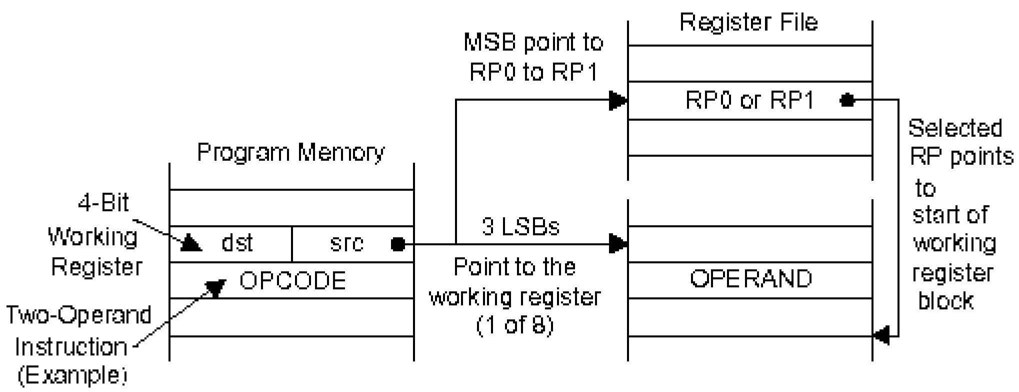
访问方法与它们的区别如下图：



程序例子：

DEC　　CNT　　　　　　;CNT 是 8 位寄存器地址的标志符号

图 3-1　　寄存器访问模式

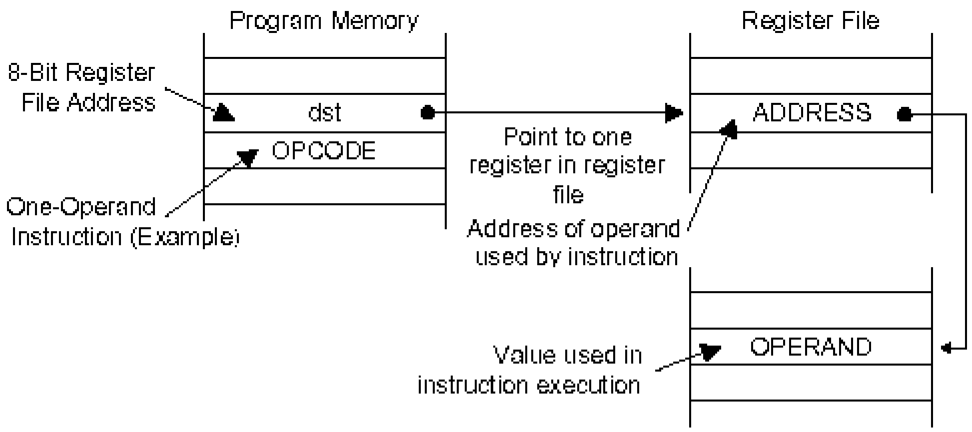


指令例子：　ADD　　R1,R2　　　　;R1,R2 是当前工作寄存器区的寄存器

图 3-2　工作寄存器访问模式

间接寄存器访问模式

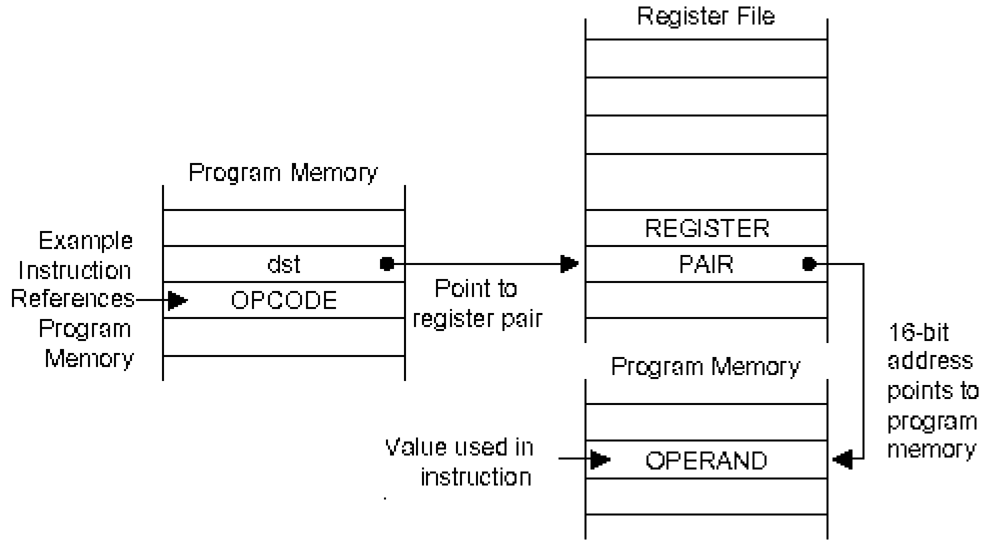
在寄存器间接访问模式中，指定寄存器或者寄存器对中存放的是操作数的地址。根据所用的指令，物理地址有可能为寄存器卷中的寄存器，程序存贮器，或者外部数据存贮器（如图 3-3 到 3-6）。你可以用任意的 8 位寄存器访问其它的寄存器，也可以用任意的 16 位寄存器组访问其它的存贮器空间。



指令例子：

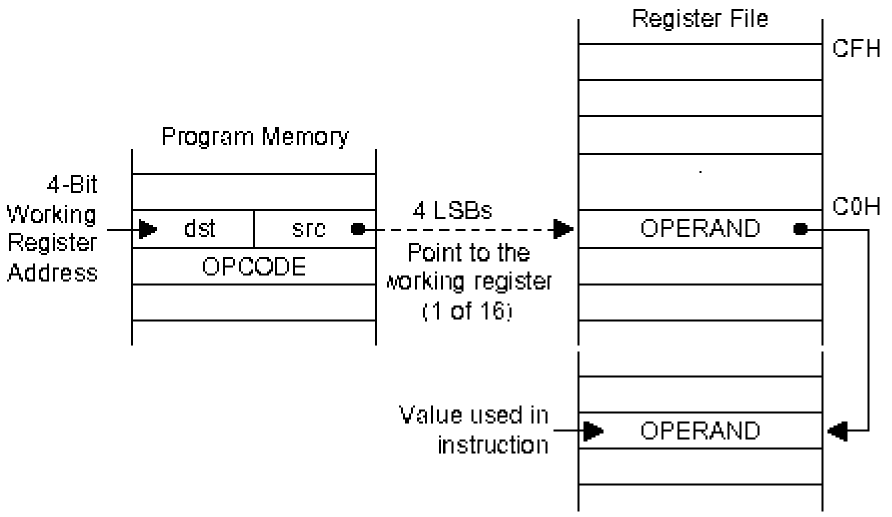
RL @SHIFT ;SHIFT 是 8 位寄存器地址的标志符号

图 3-3 寄存器卷中的间接地址访问



指令例子： CALL $@RR_2$ JP $@RR_2$

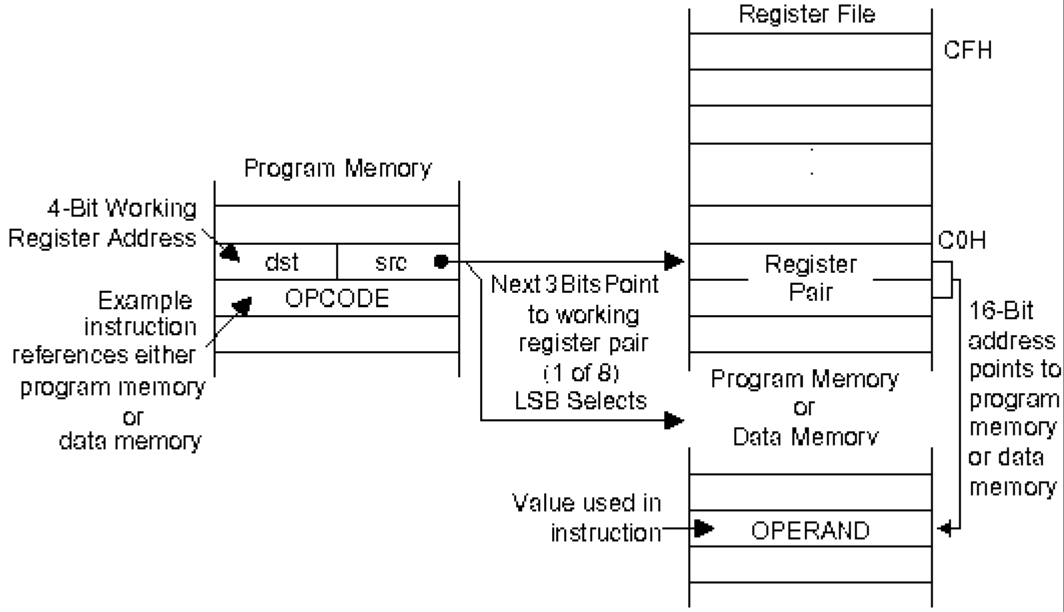
图 3-4 程序存贮空间的间接地址访问



指令例子： OR R6,@R2

图 3-5 寄存器卷中的间接地址访问

（续）



指令例子：

LDC	R5,@RR_6	；程序存贮空间访问
LDE	R3,@RR_{14}	；外部数据存贮空间访问
LDE	@RR_4,R8	；外部数据存贮空间访问

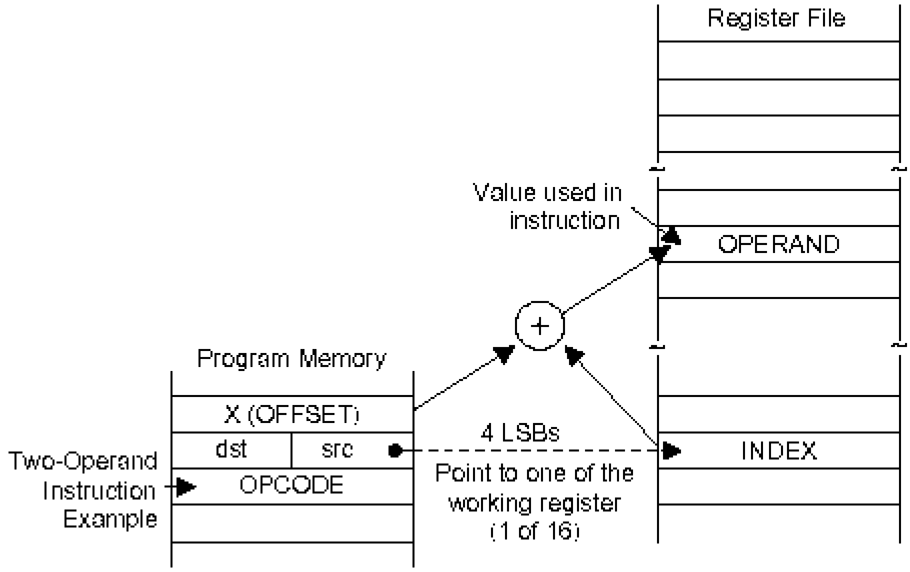
图 3-6　　工作寄存器间接访问程序存贮器和数据存贮器

偏址访问模式

在指令执行时，偏址访问模式是在基地址的基础上加上偏移地址计算出有效的操作数地址（如图 3-7）。编程时，可以利用偏址访问模式访问内部寄存器卷或者外部数据存贮器。

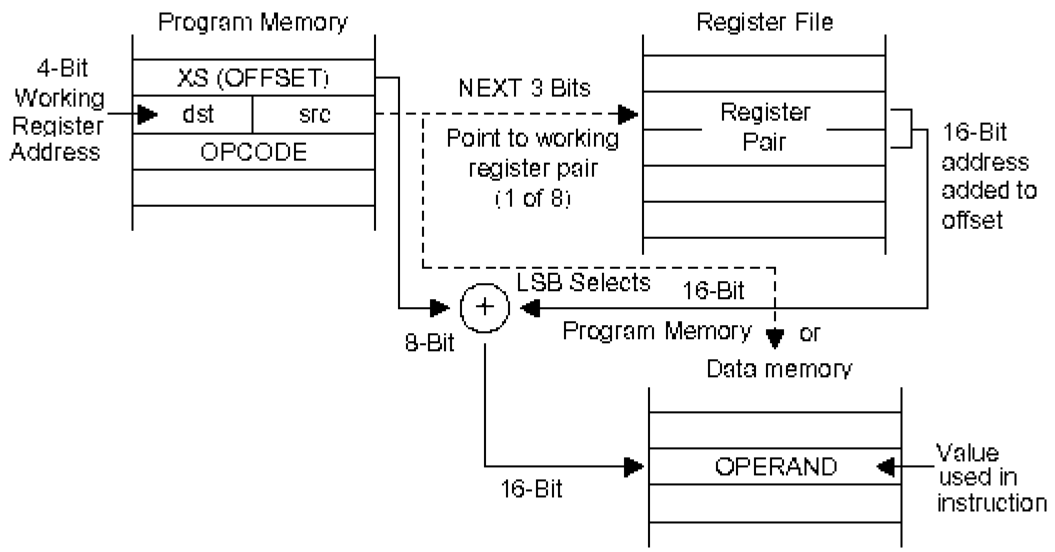
在短指令访问模式下，8 位偏移量被认为是范围在-128--+128 的一个有符号偏移量。这只用于外部存贮器访问（如图 3-8）。

对寄存器卷访问时，指令中提供的 8 位基地址加上工作寄存器中的 8 位偏移地址，进行偏址访问的。对外部存贮器访问时，基地址存放在指令指示的 16 位工作寄存器中，指令中给出的 8 位或 16 位偏移地址就加在基地址上，实现操作数访问的（如图 3-9）。



指令例子：　LD　　R0,#BASE[R1]　　　；BASE 是一个 8 位的立即数

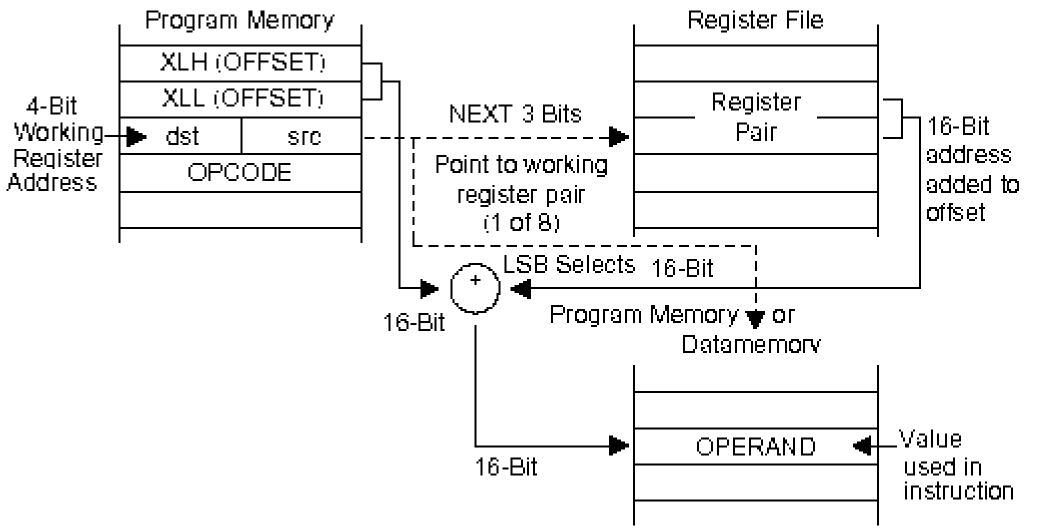
图 3-7　　　寄存器卷的偏址访问



指令例子：LDC R4,#04H[RR2] ;程序存贮空间地址(RR2+#04H)中的数送入 R4

LDE R4,#04H[RR2] ;数据存贮空间地址(RR2+#04H)中的数送入 R4

图 3-8 偏址访问模式中短格式访问程序存贮空间或数据存贮空间



指令例子：LDC R4,#1000H[RR2] ;程序存贮空间地址(RR2+#1000H)中的数送入 R4

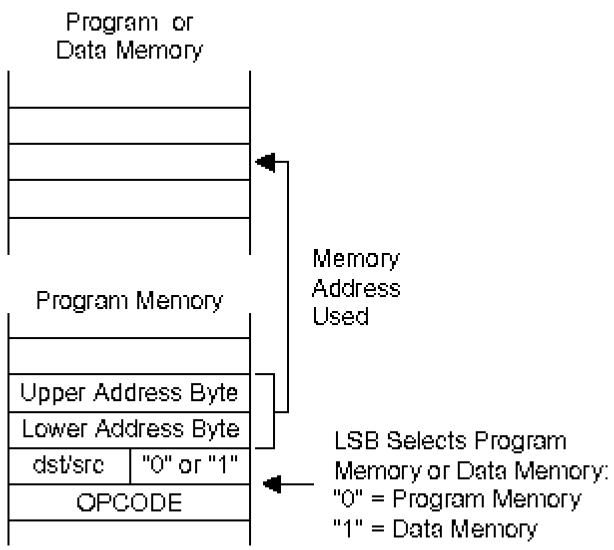
LDE R4,#1000H[RR2] ;程序存贮空间地址(RR2+#1000H)中的数送入 R4

图 3-9 偏址访问模式中长格式访问程序存贮空间或数据存贮空间

直接访问模式：

在直接访问模式下，指令提供操作数的 16 位存贮器地址。执行 JP,CALL 指令时，就是采用这种地址访问模式指定装入 PC 当中的 16 位地址。

LDC,LDE 指令即运用直接访问模式为传送数据操作提供原操作数和目的操作数的地址，LDC 访问程序存贮空间，LDE 访问外部数据存贮空间。

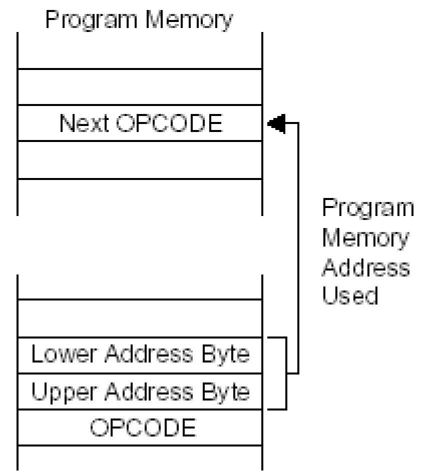


指令例子：

LDC　R5,1234H　;程序存贮空间地址为 1234H 单元的值,送入寄存器 R5

LDE　R5,1234H　;数据存贮空间地址为 1234H 单元的值,送入寄存器 R5

图　3-10　直接地址访问模式



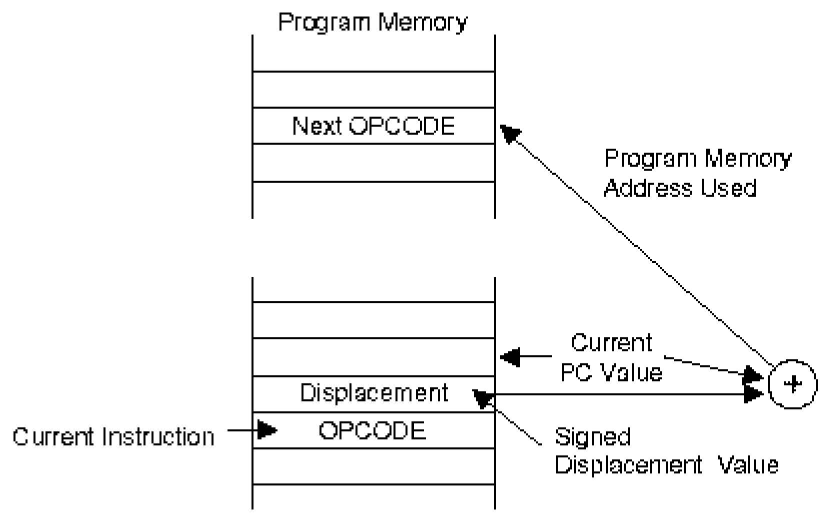
指令例子：　JP　　C , JOB1　　;JOBI 是 16 位地址
　　　　　　CALL　DISPLAY　　;DISPLAY 是 16 位地址

图　3-11　　CALL ,JP 的直接地址访问

相对地址访问模式

在相对地址访问模式中，指令的调转范围只能在有符号数-128 到+127 之间。偏移量加上当前 PC 值，即为下一条要执行指令的地址。

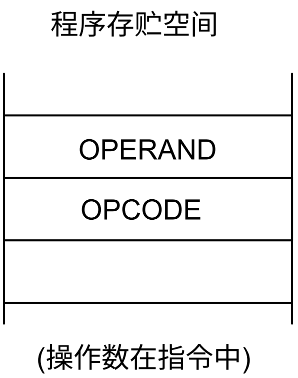
支持相对地址访问的指令是 JR。(如下图 3-12　相对地址访问)



指令例子：　JR　　ULT,$+OFFSET　;OFFSET 的范围为+127~~-128 之间

立即数访问模式（IM）

在立即数访问模式中，指令本身就已经提供了操作数的值。直接地址访问在向寄存器中装入常数时是很有用的。



指令例子：　LD　R0,#0AAH　;

图 3-13　立即数访问模式

第四章　　　控制寄存器

概述

在这一章，我们用易读的表格形式详细的描述了 S3C9444/C9454 的控制寄存器。这些信息将会帮助设计人员熟悉控制寄存器在寄存器卷中的位置，也可以用作编程人员的快速参考资源。

表 4-1 总结了系统控制寄存器和外围接口控制寄存器，插图 4-1 图解了标准寄存器描述的特征。控制寄存器描述按照寄存器代表符号的字母顺序排列。更多有关控制寄存器的信息在本手册第二部分的硬件资源描述中。

寄存器	标号	地址		复位值							
		Address	R/W	7	6	5	4	3	2	1	0
定时器 0 计数器(Timer0)	T0CNT	D0H	R	0	0	0	0	0	0	0	0
定时器 0 数据寄存器(T0)	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
定时器 0 控制寄存器(T0)	T0CON	D2H	R/W	0	0	-	-	0	-	0	0
D3H 保留											
时钟控制寄存器（Clock）	CLKCON	D4H	R/W	0	-	-	0	0	-	-	-
系统标志寄存器（Flags）	FLAGS	D5H	R/W	x	x	x	x	-	-	-	-
D6H—D8H 保留											
堆栈寄存器（SP）	SP	D9H	R/W	x	x	x	x	x	x	x	x
DAH 保留											
MDS 特殊功能寄存器	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
基 本 定 时 控 制 器 (Basic T)	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
基本定时计数器（B·T）	BTCNT	DDH	R	0	0	0	0	0	0	0	0
测 试 控 制 寄 存 器 (Test mode)	FTSTCON	DEH	W	-	-	0	0	0	0	0	0
系统模式控制寄存器	SYM	DFH	R/W	-	-	-	-	-	0	0	0

表 4-1　　系统和外围接口控制寄存器

注释：1　- 没有用到，x 值不确定。

2　FTSTCON 是生产厂家测试寄存器，只有生产厂家可以使用。在正常操作中，该寄存器的值应始终为“00H”

（续）

寄存器	标号	地址	R/W	复位值							
				7	6	5	4	3	2	1	0
P0 口数据寄存器(Port 0)	P0	E0H	R/W	0	0	0	0	0	0	0	0
P1 口数据寄存器(Port 1)	P1	E1H	R/W	-	-	-	-	-	0	0	0
P2 口数据寄存器(Port 2)	P2	E2H	R/W	-	0	0	0	0	0	0	0
E3H-E5H 保留											
P0 口 控 制 寄 存 器 (P0 High byte)	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
P0 口 控 制 寄 存 器 (P0 Low byte)	P0CONL	E7H	R/W	0	0	0	0	0	0	0	0
P0 口中断响应寄存器 (Port 0)	P0PND	E8H	R/W	-	-	-	-	0	0	0	0
P1 口控制寄存器	P1CON	E9H	R/W	0	0	-	-	0	0	0	0
P2 口 控 制 寄 存 器 (P2 High Byte)	P2CONH	EAH	R/W	-	0	0	0	0	0	0	0
P2 口 控 制 寄 存 器 (P2 Low Byte)	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
ECH—F1H 保留											
PWM 数据寄存器	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM 控制寄存器	PWMCON	F3H	R/W	0	0	-	0	0	0	0	0
STOP 控制寄存器	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
F5H—F6H 保留											
A/D 控制寄存器	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D 转换数据寄存器(High)	ADDATAH	F8H	R	X	X	X	X	X	X	X	X
A/D 转换数据寄存器 (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	0	0
FAH—FFH 保留											

注释：

1 -：没有用到，X：不确定

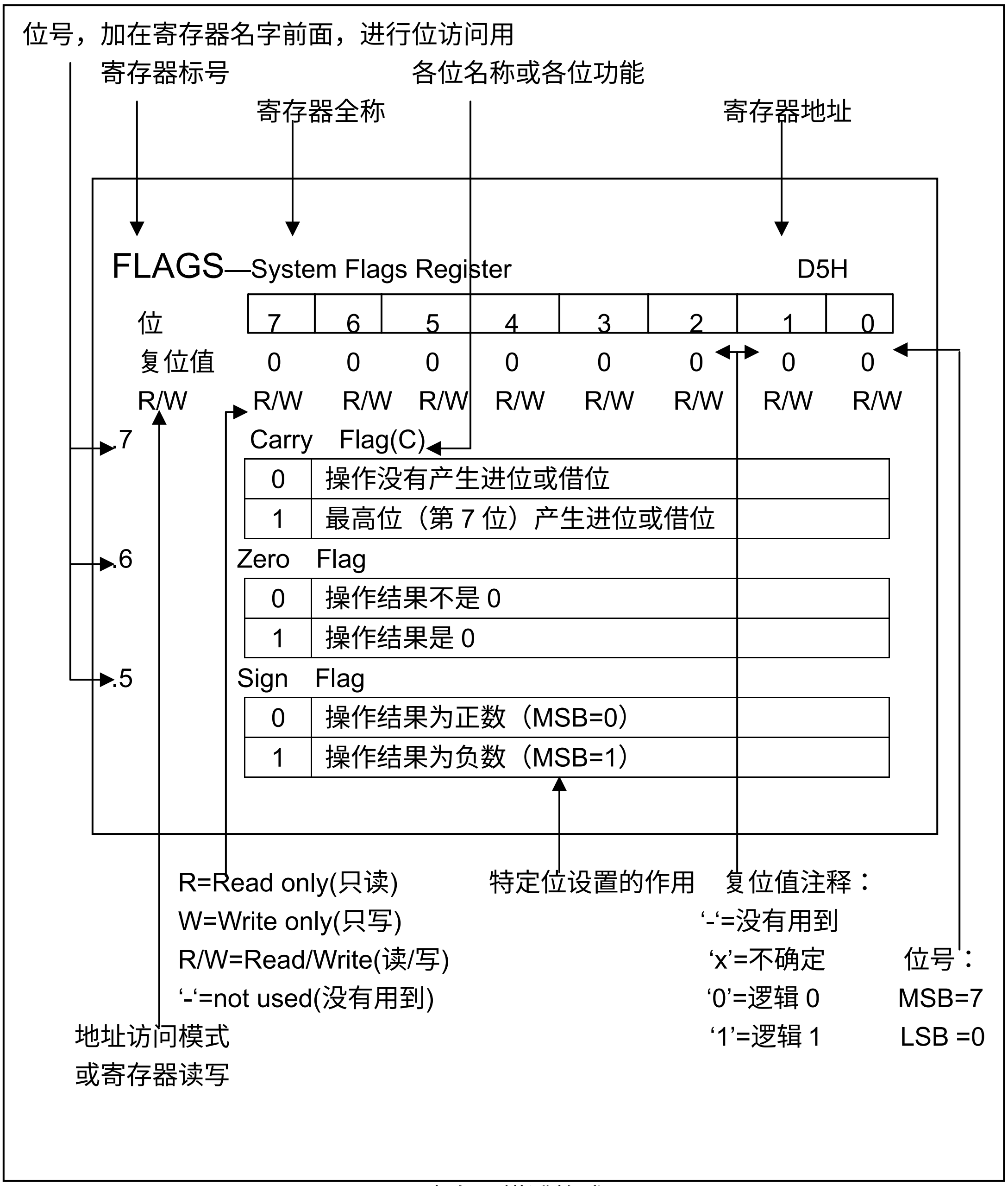

位号，加在寄存器名字前面，进行位访问用
寄存器标号
各位名称或各位功能
寄存器全称
寄存器地址
FLAGS—System Flags Register
D5H
位　7　6　5　4　3　2　1　0
复位值　0　0　0　0　0　0　0　0
R/W　R/W　R/W　R/W　R/W　R/W　R/W　R/W　R/W
.7　Carry　Flag(C)
0　操作没有产生进位或借位
1　最高位（第 7 位）产生进位或借位
.6　Zero　Flag
0　操作结果不是 0
1　操作结果是 0
.5　Sign　Flag
0　操作结果为正数（MSB=0）
1　操作结果为负数（MSB=1）
R=Read only(只读)
W=Write only(只写)
R/W=Read/Write(读/写)
‘-‘=not used(没有用到)
特定位设置的作用
复位值注释：
‘-‘=没有用到
‘x’=不确定
‘0’=逻辑 0
‘1’=逻辑 1
位号：
MSB=7
LSB =0
地址访问模式
或寄存器读写

图 4-1　　寄存器描述格式

ADCON—A/D Converter Control Register　　F7H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.4　A/D 转换输入引脚选择位

0	0	0	0	ADC0 (P0.0)
0	0	0	1	ADC1 (P0.1)
0	0	1	0	ADC2 (P0.2)
0	0	1	1	ADC3 (P0.3)/在 S3C9444 中，内部接地
0	1	0	0	ADC4 (P0.4)/在 S3C9444 中，内部接地
0	1	0	1	ADC5 (P0.5)/在 S3C9444 中，内部接地
0	1	1	0	ADC6 (P0.6)/在 S3C9444 中，内部接地
0	1	1	1	ADC7 (P0.7)/在 S3C9444 中，内部接地
1	0	0	0	ADC8 (P2.6)/在 S3C9444 中，内部接地
1	0	0	1	内部接地
1	0	1	0	内部接地
1	0	1	1	内部接地
1	1	0	0	内部接地
1	1	0	1	内部接地
1	1	1	0	内部接地
1	1	1	1	内部接地

.3　转换结束状态位

0	A/D 转换正在进行
1	A/D 转换结束

.2-.1　时钟选择位

0	0	fosc/16(fosc≤10MHZ)
0	1	fosc/8(fosc≤10MHZ)
1	0	fosc/4(fosc≤10MHZ)
1	1	fosc/1(fosc≤2.5MHZ)

.0　启动转换位

0	没有意义
1	启动 A/D 转换

BTCON—Basic Timer Control Register　　DCH

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.4　看门狗功能允许位

1	0	1	0	禁止看门狗功能
其 它 值				允许看门狗功能

.3--.2　Basic Timer 输入时钟选择位

0	0	fosc/4096
0	1	fosc/1024
1	0	fosc/128
1	1	非法设置

.1　Basic Timer 清 0 控制位

0	没有作用
1	清除 Basic Timer 的计数值

.0　Basic Timer 分频器清除位

0	没有作用
1	清除分频器

注意：当写'1'到 BTCON.0(或 BTCON.1)时，Basic Timer 立即清除计数值（或分频器），同时该位也被立即清除。

CLKCON—Clock Control Register D4H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	—	—	0	0	—	—	—
R/W	R/W	—	—	R/W	R/W	—	—	—

.7 晶振 IRQ 唤醒功能使能位

0	允许 IRQ 唤醒系统主晶振
1	禁止 IRQ 唤醒系统主晶振

.6--.5

在 S3C9444/C9454 中没有用到

.4--.3 CPU 时钟分频选择位

0	0	16 分频(fosc/16)
0	1	8 分频(fosc/8)
1	0	2 分频(fosc/2)
1	1	1 分频(fosc)

.2--.0

在 S3C9444/C9454 中没有用到

FLAGS—System Flags Register　　D5H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—

.7 Carry Flag(C)

0	操作没有产生进位或借位
1	操作产生进位或借位

.6 Zero Flag(Z)

0	操作结果不是“0”
1	操作结果是“0”

.5 Sign Flag(S)

0	操作产生正数（MSB=”0”）
1	操作产生负数（MSB=”1”）

.4 Overflow Flag(V)

0	操作结果在-128~+127 之间
1	操作结果不在-128~+127 之间，即溢出

.3--.0

在 S3C9444/C9454 中没有用到

P0CONH—Port 0 Control Register (High Byte) E6H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.6　Port 0,P0.7/INT7 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入，不带上拉电阻
1	0	推挽式输出
1	1	A/D 转换输入（ADC7)，施密特触发器输入关闭

.5--.4　Port0,P0.6/ADC6/PWM 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	相关功能（PWM 输出）
1	0	推挽式输出
1	1	A/D 转换输入（ADC6)，施密特触发器输入关闭

.3--.2　Port0,P0.5/ADC5 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	A/D 转换输入（ADC5)，施密特触发器输入关闭

.1--.0　Port0,P0.4/ADC4 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	A/D 转换输入（ADC4)，施密特触发器输入关闭

P0CONL—Port0 Control Register (Low Byte)　E7H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.6　Port 0,P0.3/INT3 功能位

0	0	施密特触发器输入
0	1	施密特触发器输入，带上拉电阻
1	0	推挽式输出
1	1	A/D 转换输入（ADC3），施密特触发器输入关闭

.5--.4　Port 0,P0.2/ADC2 功能位

0	0	施密特触发器输入
0	1	施密特触发器输入，带上拉电阻
1	0	推挽式输出
1	1	A/D 转换输入（ADC2），施密特触发器输入关闭

.3--.2　Port 0,P0.1/ADC1/INT1 功能位

0	0	施密特触发器输入/下降沿产生中断
0	1	施密特触发器输入，带上拉电阻/下降沿产生中断
1	0	推挽式输出
1	1	A/D 转换输入（ADC1），施密特触发器输入关闭

.1--.0　Port 0,P0.0/ADC0/INT0 功能位

0	0	施密特触发器输入下降沿产生中断
0	1	施密特触发器输入，带上拉电阻/下降沿产生中断
1	0	推挽式输出
1	1	A/D 转换输入（ADC0），施密特触发器输入关闭

P0PND—Port 0 Interrupt Pending Register　E8H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	—	—	—	—	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

.7--.4　在 S3C9444/C9454 中没有用到

.3　P0.1/ADC1/INT1 中断使能位

0	INT1 禁止下降沿中断
1	INT1 允许下降沿中断

.2　P0.1 /ADC1/INT1 中断标志位

0	没有中断（读此位时，如果是 0 则没有中断）
0	中断标志清除（当向此位写 0 时，则清除中断标志）
1	中断标志位置起（读此位时，如果是 1 则有中断）
1	没有作用（当向此位写 1 时）

.1　P0.0/ADC0/INT0 中断使能位

0	INT0 禁止下降沿中断
1	INT0 允许下降沿中断

.0　P0.0/ADC0/INT0 中断标志位

0	没有中断（读此位时，如果是 0 则没有中断）
0	中断标志清除（当向此位写 0 时，则清除中断标志）
1	中断标志位置起（读此位时，如果是 1 则有中断）
1	没有作用（当向此位写 1 时）

P1CON—Port1 Control Register E9H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	—	—	0	0	0	0
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W

.7 Port1.1 N 沟道开漏输出使能位

0	P1.1 用作推挽式输出
1	P1.1 用作 N 沟道开漏输出

.6 Port1.0 N 沟道开漏输出使能位

0	P1.0 用作推挽式输出
1	P1.0 用作 N 沟道开漏输出

.5--.4 在 S3C9444/C9454 中没有用到

.3--.2 Port 1,P1.1 功能位

0	0	施密特触发器输入
0	1	施密特触发器输入，带上拉电阻
1	0	输出
1	1	施密特触发器输入，带下拉电阻

.1--.0 Port1,P1.0 功能位

0	0	施密特触发器输入
0	1	施密特触发器输入，带上拉电阻
1	0	输出
1	1	施密特触发器输入，带下拉电阻

注释：当用外部晶振时，P1.0,P1.1 必须设置为输出模式，以减少电流消耗。

P2CONH—Port 2 Control Register (High Byte)　　EAH

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	—	0	0	0	0	0	0	0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7　在 S3C9444/C9454 中没有用到

.6--.4　Port2,P2.6/ADC8/CLO 功能位

0	0	0	施密特触发器输入，带上拉电阻
0	0	1	施密特触发器输入
0	1	x	ADC 输入
1	0	0	推挽式输出
1	0	1	开漏输出，带上拉电阻
1	1	0	开漏输出
1	1	1	相关功能，CLO 输出

.3--.2　Port2,P2.5 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	开漏输出

.1--.0　Port2,P2.4 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	开漏输出

注释：当噪声比较严重时，最好不要用 CLO 输出。

P2CONL—Port2 Control Register (LowByte) EBH

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.6 Port2，P2.3 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	开漏输出

.5--.4 Port2，P2.2 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	开漏输出

.3--.2 Port2，P2.1 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	开漏输出

.1--.0 Port2，P2.0 功能位

0	0	施密特触发器输入，带上拉电阻
0	1	施密特触发器输入
1	0	推挽式输出
1	1	T0 匹配输出

PWMCON—PWM Control Register E3H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	—	0	0	0	0	0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W

.7--.6 PWM 时钟输入选择位

0	0	fosc/64
0	1	fosc/8
1	0	fosc/2
1	1	fosc/1

.5 在 S3C9444/C9454 中没有用到

.4 PWMDATA 重装周期选择位

0	8 位计数溢出之后重装
1	6 位计数溢出之后重装

.3 PWM 计数清零位

0	没有作用
1	清除 PWM 计数值，（当向其斜的时候）

.2 PWM 计数使能位

0	停止计数
1	开始计数

.1 PWM 溢出中断使能位

0	禁止中断
1	允许中断

.0 PWM 溢出中断标志位

0	没有中断(当读此位时，如果为 0，则表示没有 PWM 中断)
0	清除中断标志位（当向此位写 0 时，则清除中断标志）
1	产生中断标志（读此位时，如果为 1，则表示有 PWM 中断）
1	没有作用（当向此位写 1 时，没有作用）

注释：PWMCON.3 不自动清除，在清除中断标志位时，必须给与注意。

STOPCON—STOP Mode Control Register E4H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7--.0 看门狗定时器功能使能位

10100101	允许用 STOP 指令
其他值	禁止用 STOP 指令

注释：如果 STOPCON 寄存器不是 A5H，当指令中用 STOP 指令时，PC 值将会调到复位地址（0100H）。

SYM — System Mode Register DFH

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	—	—	—	—	—	0	0	0
R/W	—	—	—	—	—	R/W	R/W	R/W

.7--.3 在 S3C9444/C9454 中没有用到

.2 全局中断使能位

0	禁止所有中断
1	允许所有中断

.1--.0 页选择位

0	0	第 0 页
0	1	第 1 页（在 S3C9444/C9454 中没有用到）
1	0	第 2 页（在 S3C9444/C9454 中没有用到）
1	1	第 3 页（在 S3C9444/C9454 中没有用到）

T0CON—TIMER0 Control Register F4H

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	—	—	0	—	0	0
R/W	R/W	R/W	—	—	R/W	—	R/W	R/W

.7--.6 定时器 0（TIMER 0）时钟输入选择位

0	0	fosc/4096
0	1	fosc/256
1	0	fosc/8
1	1	fosc/1

.5--.4 在 S3C9444/C9454 中没有用到

.3 定时器 0 计数清 0 位

0	没有作用
1	清除定时器 T0 的计数值（当向此位写 1 时，则清除此位计数值）

.2 在 S3C9444/C9454 中没有用到

.1 定时器 T0 中断使能位

0	禁止定时器中断
1	允许定时器中断

.0 定时器中断标志位（捕捉中断或相等中断）

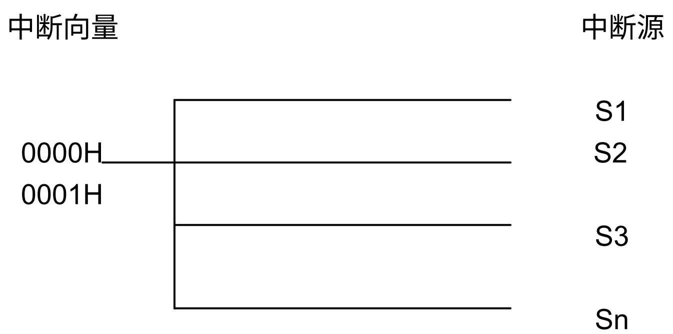
0	没有中断(当读此位时，如果为 0，则表示没有定时器中断)
0	清除中断标志位（当向此位写 0 时，则清除中断标志）
1	产生中断标志（读此位时，如果为 1，则表示有定时器中断）
1	没有作用（当向此位写 1 时，没有作用）

注释：
1. 当清除中断标志位时，要注意 T0CON.3,因为该位不自动清除。
2. 当 T0 工作于定时模式时，应先把 T0CON.3 清 0。

第五章 中断

概述

SAM88RCRI 的中断结构有两个组成部分：中断向量和中断源。中断源可以通过 ROM 地址为 0000H 的中断向量得到 CPU 的响应。



注释： 1 SAM88RCRI 只有一个中断向量地址（0000H）

2 Sn 是扩展的

图 5—1 S3F9 系列的中断类型

中断处理的控制

中断控制可以通过两种方法处理：全局中断控制和具体中断、中断源的控制。

---全局中断的允许与禁止。EI 指令允许全局中断，DI 指令禁止全局中断。

---中断源的允许与禁止。通过设置相应的外围接口控制寄存器可以允许或禁止相应的中断。

系统模式寄存器（SYM）用于允许或禁止中断处理。SYM.3 是允许或禁止中断处理的标志位。在复位操作之后，为了允许中断处理，应该在初始化程序中打开中断。尽管可以直接在 SYM.3 位写“1”或“0”来允许或禁止中断，**但我们还是建议用 EI 指令或 DI 指令。**

当处理中断服务程序时，中断服务程序必须在中断返回之前，清除相应的中断标志位，因为 SAM88RCRI 硬件不自动清除中断标志位，**必须由软件清除相应的中断标志位。**

中断优先级

SAM88RCRI 没有中断优先级寄存器，中断执行顺序是由响应中断后，中断服务程序处理中断的顺序决定的。

中断响应检测与服务顺序如下：

1. 产生中断请求，在相应中断标志位置“1”。
2. CPU 响应中断请求。
3. 中断服务程序处理，软件清除相应中断标志位。
4. 软件检测相应中断标志位决定中断优先级。

在中断响应以前，必须满足下列条件：

--- 必须允许中断处理（EI，SYM.2=“1”）。

--- 设置相应中断控制寄存器的有关位。

如果上述条件都已满足，CPU 在最后一个指令周期响应中断请求，开始中断处理完成下面操作：

1. 清除中断允许位 SYM.3(DI,SYM.3=“0”),禁止所有后来的中断。
2. 把 PC 值和系统标志寄存器（FLAGS）压入堆栈。
3. 转到中断向量地址，把中断服务程序地址压入 PC。
4. 处理中断服务程序

当处理完中断服务程序之后，中段返回指令（IRET）弹出 PC 值和系统标志寄存器的值至 PC、FLAGS 中，同时置起 SYM.2(SYM.2=“1”)，允许 CPU 继续处理中断。

ROM 中的中断向量地址中存放着中断服务程序的地址。中断处理的顺序如下：

1. 把 PC 低字节压入堆栈
2. 把 PC 高字节压入堆栈
3. 把 FLAGS 寄存器的值压入堆栈
4. 从中断向量地址 0000H 中取出中断服务程序高字节地址
5. 从中断向量地址 0001H 中取出中断服务程序低字节地址
6. 转入执行 16 位中断向量地址所确定的中断服务程序

S3C9444/C9454 中断结构

S3C9444/C9454 有 4 个外围中断源

---- PWM 溢出中断

---- T0 匹配中断

---- P0.0 外部中断

---- P0.1 外部中断

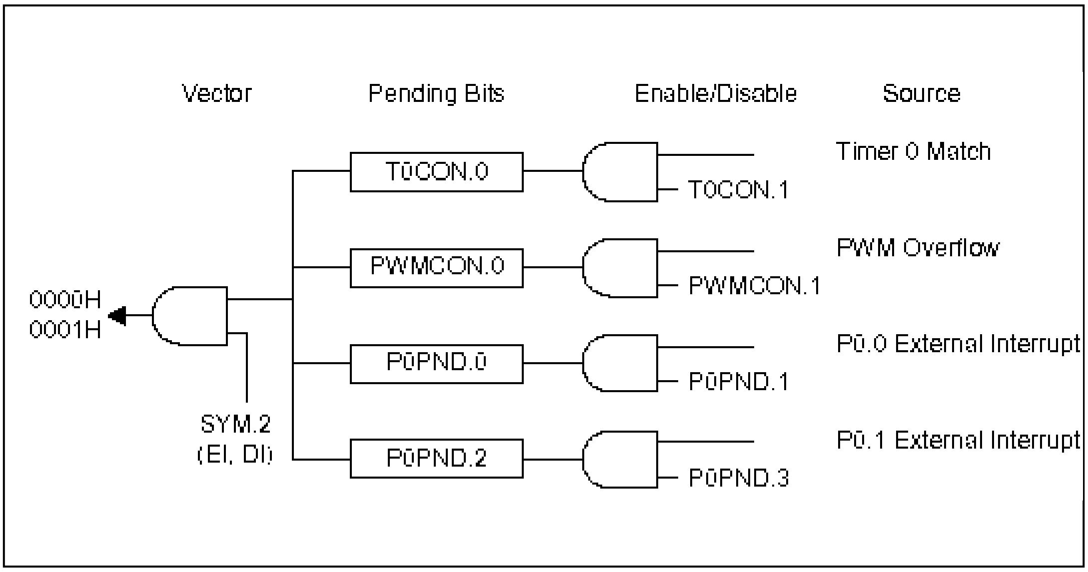


图 5-3　　S3C9444/C9454 中断源

第六章　　SAM88RCRI 指令集

概述

SAM88RCRI 指令集支持寄存器卷操作，它可完成 8 位算术操作和逻辑操作，共有 41 条指令集。由于采用了存贮器影射方式，所以没有具体的 I/O 口操作指令。SAM88RCRI 指令集提供强大的数据处理能力，支持位寻址、循环、移位等数据操作。

为访问个别寄存器，应指定寄存器卷中 0--255 的 8 位地址或工作寄存器中的 4 位地址。工作寄存器中，寄存器对可以访问 13 位程序存贮空间和数据存贮空间。SAM88CRI 支持 6 种地址访问方法，方便了编程操作。

指令简介

数据传送类指令

CLR	dst	清 0
LD	dst,src	传输数据
LDC	dst,src	传送数据（访问程序存贮空间）
LDE	dst,src	传送数据（访问数据存贮空间）
LDCD	dst,src	传送数据后地址减 1（访问程序存贮空间）
LDED	dst,src	传送数据后地址减 1（访问数据存贮空间）
LDCI	dst,src	传送数据后地址加 1（访问程序存贮空间）
LDEI	dst,src	传送数据后地址加 1（访问数据存贮空间）
POP	dst,src	弹栈
PUSH	dst,src	压栈

算术操作

ADC	dst,src	带进位加法
ADD	dst,src	不带进位加法
CP	dst,src	比较指令
DEC	dst	减 1 指令
INC	dst	加 1 指令
SBC	dst,src	带进位减法
SUB	dst,src	不带进位减法

逻辑操作

AND	dst,src	逻辑与
COM	dst	取反
OR	dst,src	逻辑或
XOR	dsr,src	逻辑异或

程序控制类指令

CALL	dst	调子程序
IRET		中断返回
JP	cc,dst	有条件跳转
JP	dst	无条件跳转
JR	cc,dst	有条件相对跳转
RET		子程序返回

位操作指令

TCM	dst,src	取反而后测试
TM	dst,src	测试指令

循环移位指令

RL	dst	循环左移
RLC	dst	带进位循环左移
RR	dst	循环右移
RRC	dst	带进位循环右移
SRA	dst	算术左移

CPU 控制类指令

CCF	C 取反
DI	禁止中断
EI	允许中断
IDLE	空闲指令
NOP	空操作
RCF	C 清 0
SCF	C 置 1
STOP	冻结指令

标志寄存器（FLAGS）

8 位标志寄存器描述当前 CPU 的操作状态。其中的 4 位（FLAGS.4-FLAGS.7）可以测试并用于条件转移指令。FLAGS 寄存器是可以读写操作的，数据传送类指令并不影响标志寄存器，逻辑算术操作如 AND、OR、XOR、ADD、SUB 则影响标志寄存器。例如，与操作将会影响 Z、S、O 标志寄存器，标志寄存器的状态有操作结果决定。如果与指令的目的操作数是标志寄存器，则将会两次对标志寄存器进行写操作，其结果将是不确定的，最好不要把标志寄存器作为目的操作数。

标志寄存器描述

溢出标志（FLAGS.4，V）

当操作结果大于+127 小于-128 时，溢出标志将会置 1。逻辑操作之后，它将会被清 0。

符号标志位（FLAGS.5，S）

算术、逻辑、循环、移位操作之后，符号位标志操作结果最高位的状态。逻辑 0 表示操作结果是正数，逻辑 1 表示操作结果是负数。

零标志位（FLAGS.6，Z）

如果算术、逻辑操作的结果为 0，则此标志位为 1。位操作指令、移位指令、循环移位指令都会影响此标志位。

进（借）位标志位（FLAGS.7，C）

如果算术操作后，最高位产生进位或借位，则此标志位为 1。移位、循环移位操作之后，此位保存的值为最后溢出的值。指令可以对此位置 1、清 0、取反操作。

表 6-2　　　　标志位符号

标 志 位	特 性 描 述
C	进（借）位标志位
Z	零标志位
S	符号标志位
V	溢出标志
0	清为逻辑 0
1	置为逻辑 1
*	根据相应操作置 1 或清 0
--	不受影响
×	不确定

表 6-3 指令系统标号

标 号	特 性 描 述
dst	目的操作数
src	源操作数
@	间接寄存器寻址前缀
PC	程序计数器
FLAGS	状态寄存器
#	立即数或寄存器访问的前缀
H	16 进制数后缀
D	10 进制数后缀
B	二进制数后缀
opc	操作代码

表 6-4 指令符号的定义

符号	描述	实际操作范围
cc	条件转移代码	参考表 6-6
r	工作寄存器	Rn(n=0-15)
rr	工作寄存器对	RRp(p=0,2,4,…..,14)
R	寄存器或工作寄存器	reg 或 Rn(reg=0-255,n=0-15)
RR	寄存器对或工作寄存器对	Reg 或 RRp(reg=0-254,p=0,2,4,….,14)
Ir	间接地址访问工作寄存器	@Rn(n=0-15)
IR	间接地址访问寄存器或工作寄存器	@Rn 或@reg(reg=0-254,p=0,2,4,….,14)
Irr	间接地址访问工作	@RRp(p=0,2,4,…,14)
IRR	间接寄存器对或工作寄存器对访问	@RRp 或@reg(reg=0-254,p=0,2,4,…,14)
X	偏址访问模式	#reg[Rn](reg=0-255,n=0-15)
XS	短偏址访问模式	#addr[RRp](addr=-128—127,p=0,2,4,…,14)
XL	长偏址访问模式	#addr[RRp](addr=0-8191, p=0,2,4,…,14)
DA	直接地址访问模式	addr(addr=0-8191)
RA	相对地址访问模式	addr(addr=-128--127
IM	立即数访问模式	#data(data=0-255)

表 6-5 操作代码参考表

OPCODE MAP									
LOWER NIBBLE(HEX)									
	-	0	1	2	3	4	5	6	7
UPPER NIBBER HEX	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD r1,r2	ADD IR1,r2	ADD R1,IM	
	1	RLC R1	RLC IR1	ADD r1,r2	ADD r1,Ir2	ADD r2,r1	ADD Ir2,r1	ADD R1,IM	
	2	INC R1	INC IR1	SUB r1,r2	SUB R1,IR2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	
	3	JP IRR1		SBC r1,r2	SBC R1,IR2	SBC R2,R1	SBC IR_2,R_1	SBC R1,IM	
	4			OR RI,R2	OR R1,IR2	OR R2,R1	OR IR_2,R_1	OR R1,IM	
	5	POP R1	POP IR1	AND RI,R2	AND R1,IR2	AND R2,R1	AND IR_2,R_1	AND R1,IM	
	6	COM R1	COM IR1	TCM R1,R2	TCM R1,IR2	TCM R2,R1	TCM IR_2,R_1	TCM R1,IM	
	7	PUSH R2	PUSH IR2	TM R1,R2	TM R1,IR2	TM R2,R1	TM IR_2,R_1	TM R1,IM	
	8								LD R1,X,R2
	9	RL r1	RL IR1						LD R2,X,R1
	A			CP R1,R2	CP R1,IR2	CP R2,R1	CP IR_2,R_1	CP R1,IM	LDC R1,IRR2,XL
	B	CLR R1	CLR IR1	XOR R1,R2	XOR R1,IR2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC R2,IRR2,XL
	C	RRC R1	RRC IR1		LDC R1,IRR2				LD R1,IR2
	D	SRA R1	SRA IR1		LDC R2,IRR1			LD IR1,IM	LD IR1,R2
	E	RR R1	RR IR1	LDCD R1,IRR2	LDCI R1,IRR2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC R1,IRR2,XS
	F					CALL IRR1	LD IR2,R1	CALL DA1	LDC R2.IRR1,XS

表 6-5　　操作代码参考表

OPCODE MAP									
LOWER NIBBLE (HEX)									
	-	8	9	A	B	C	D	E	F
U	0	LD	LD		JR	LD	JP	INC	
P	1	R1,R2	R2,R1		cc,RA	R1,IM	cc,DA	R1	
P	2								
E	3								
R	4								
	5	↓	↓		↓	↓	↓	↓	
N	6								IDLE
I	7								STOP
B	8	↓	↓		↓	↓	↓	↓	DI
B	9								EI
L	A								RET
E	B	↓	↓		↓	↓	↓	↓	IRET
	C								RCF
H	D								SCF
E	E	LD	LD		JR	LD	JP	INC	CCF
X	F	R1,R2	R2,R1		cc,RA	R1,IM	cc,DA	R1	NOP

条件转移代码

条件转移指令经常包含 4 位条件转移操作判断。这些条件转移操作判断决定程序的调转方向。条件代码操作判断表如下。

C，Z，S，V 等标志位用作条件转移判断位。指令将会根据这些标志位决定调转方向。

二进制	符 号	描 述	符号位状态
0000	F	逻辑假	--
1000	T	逻辑真	--
0111 (1)	C	有进位或借位	C=1
1111 (1)	NC	无进位或借位	C=0
0110 (1)	Z	结果为 0	Z=1
1110 (1)	NZ	结果不为 0	Z=0
1101	PL	正数	S=0
0101	MI	负数	S=1
0100	OV	溢出	V=1
1100	NOV	没有溢出	V=0
0110 (1)	EQ	相等	Z=1
1110 (1)	NE	不等	Z=0
1001	GE	大于等于	（S⊕V）=0
0001	LT	小于	（S⊕V）=1
1010	GT	大于	Z|（S⊕V）=0
0010	LE	小于等于	Z|（S⊕V）=1
1111 (1)	UGE	无符号数大于等于	C=0
0111 (1)	ULT	无符号数小于	C=1
1011	UGT	无符号数大于	(C=0&Z=0)=1
0011	ULE	无符号数小于等于	(C|Z)=1

表 6-6 条件转移代码

注释：

1 条件代码虽然标志 2 个不同的标号，但测试相同的位。例如，Z，EQ 都为真，如果 Z 符号位被置起，但是 ADD 指令操作之后，Z 可能被用到；CP 指令操作之后，EQ 可能被用到。

2 如果操作涉及到无符号数，则必须用 UGE，UDT，ULT，ULE 等条件代码。

指令集介绍

这一章详细地介绍了指令操作同时列出了具体的编程实例。在介绍指令时，我们用了统一的形式，以便参阅、查找。在介绍每条指令时，我们采用了如下的描述方法：

— 指令名称（符号）
— 指令全称
— 源操作数/目的操作数的格式
— 具体指令的解释
— 具体描述了每条指令的操作
— 每条指令对标志寄存器的影响
— 详细的介绍了指令格式、执行周期和访问模式
— 给出了每条指令的编程实例

ADC　　带进位加法（Add With Carry）

ADC　　dst,src

操　作：　dst←dst + src + C

目的操作数加上源操作数和 C 位，所得结果放入目的操作数地址。源操作数不受影响。在多字节加法中，需要把进位加到下一次运算中，则多用带进位加法指令。

标志位：　C：如果加法运算中，产生进位，则此位置 1；反之，清 0。

Z: 如果运算结果为 0,该位置 1；反之，置 0。

S：如果运算结果为负，则此位置 1；反之，清 0。

V：如果运算中产生溢出，则此位置 1；反之，清 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
opc | dst|src	2	4	12	r	r
		6	13	r	Ir
opc | src | dst	3	6	14	R	R
		6	15	R	IR
opc | dst | src	3	6	16	R	IM

编程实例：　假如 R1=10H，R2=03H，C=1，寄存器 01H=20H，02H=03H，03H=0AH。

```
ADC    R1,R2        ;R1=14H,R2=03H
ADC    R1,@R2       ;R1=1BH,R2=03H
ADC    01H,02H      ;寄存器 01H=24H，寄存器 02H=03H
ADC    01H,@02H     ;寄存器 01H=2BH，寄存器 02H=03H
ADC    01H,#11H     ;寄存器 01H=32H
```

在第一个例子中，目的寄存器 R1 的值为 10H，C=1，原操作数的值为 03H。带进位加法所进行的操作是，(R1)+C+(R2)=14H,加法操作之后，源操作数不变，即（R2）不变。（R1）=14H。

ADD　　加法（ADD）

ADD　　dst,src

操　作：　dst←dst + src

源操作数加上目的操作数所得的总和放在目的操作数所在的存贮空间。源操作数不变。

标志位： C：如果加法运算中，产生进位，则此位置 1；反之，清 0。

Z：如果运算结果为 0,该位置 1；反之，置 0。

S：如果运算结果为负，则此位置 1；反之，清 0。

V：如果运算中产生溢出，则此位置 1；反之，清 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst|src	2	4	02	r	r
		6	03	r	Ir
opc | src | dst	3	6	04	R	R
		6	05	R	IR
opc | dst | src	3	6	06	R	IM

编程实例：　假如 R1=12H，R2=03H，寄存器 01H=21H，02H=03H，03H=0AH。

```
ADD    R1,R2        ;R1=15H,R2=03H
ADD    R1,@R2       ;R1=1CH,R2=03H
ADD    01H,02H      ;寄存器 01H=24H，寄存器 02H=03H
ADD    01H,@02H     ;寄存器 01H=2BH，寄存器 02H=03H
ADD    01H,#25H     ;寄存器 01H=46H
```

在第一个例子中，目的寄存器 R1 的值为 12H，原操作数的值为 03H。不带进位加法所进行的操作是，(R1)+(R2)=15H,加法操作之后，源操作数不变，即（R2）不变。(R1)=15H。

AND 逻辑与(Logic AND)

AND dst←dst & src

操 作： 目的操作数与源操作数进行"与"操作，结果放在目的操作数存贮空间。源操作数不受影响。在进行与操作时，源操作数和目的操作数相应位都为 1 时，结果为 1。反之，为 0。

标志位： C：不受影响。
Z：如果运算结果为 0,该位置 1；反之，置 0。
S：如果结果的第 7 位为 1，则该位为 1；反之，为 0。
V：该操作之后，此位为 0。

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
opc | dst|src	2	4	52	r	r
		6	53	r	Ir
opc | src | dst	3	6	54	R	R
		6	55	R	IR
opc | dst | src	3	6	56	R	IM

编程实例： 假如 R1=12H，R2=03H，寄存器 01H=21H，02H=03H，03H=0AH。

```
AND    R1,R2          ;R1=02H,R2=03H
AND    R1,@R2         ;R1=02H,R2=03H
AND    01H,02H        ;寄存器 01H=01H，寄存器 02H=03H
AND    01H,@02H       ;寄存器 01H=00H，寄存器 02H=03H
AND    01H,#25H       ;寄存器 01H=21H
```

在第一个例子中，目的寄存器 R1 的值为 12H，原操作数的值为 03H。与操作所进行的操作是，(R1)&(R2)=02H,加法操作之后，源操作数不变，即（R2）不变。(R1）=02H。

CALL 调子程序

CALL dst

操 作：

SP←SP – 1
@SP←PCL
SP ←SP – 1
@SP ←PCH
PC←dst

当前 PC 计数器的值被压入堆栈。PC 值是 CALL 指令下的指令地址，即 CALL 指令下面的指令地址被压入 PC 中。之后，子程序第一条指令地址弹到 PC 计数器中，CPU 开始执行子程序。RET 指令把堆栈中存放的值，弹出到 PC 中。程序继续向下执行。

标志位： 标志寄存器不受影响。

格 式：

格式		字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc	dst	3	14	F6	DA
opc	dst	2	12	F4	IRR

编程实例： 假如 R0=15H，R1=21H，PC=1A47H，SP=0B2H。

```
CALL    1521H       ；SP=0B0H
CALL    @RR0        ；SP=0B0H
```

在第一个例子中，PC 当前值为 1A47H，SP 当前值为 0B2H。执行 CALL 指令后，系统把 1AH 压入 0B0H，把 4AH 压入 0B1H，因为该指令为 3 个字节，所以压入堆栈的是 4AH，而不是 47H。SP=0B0H。CPU 执行子程序中的指令。

在第二个例子中，执行情况与第一个例子相同，只不过压入堆栈的是 1AH 和 49H，因为这条指令占用 2 个字节，而不是 3 个字节。

CCF C 取反

CCF

操 作： C←C 取反

对 C 取反。如果 C 为 1，则此操作之后，C 为 0；反之，为 1。

标志位： C：C 取反

其它标志位不受影响。

格 式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	EF

编程实例： 假如 C=0。

CCF

如果 C=0，执行此指令后，C=1，即对 C 取反。

CLR 清 0

CLR dst

操 作： dst ← 0

目的操作数存贮器被清 0。

标志位： 该操作不影响标志位。

格 式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	B0	R
		4	B1	IR

编程实例： 假如寄存器 00H=4FH，01H=02H，02H=5EH

```
CLR    00H      ；00H=0
CLR    @01H     ；01H=02H，02H=0
```

在寄存器访问模式下，把寄存器 00H 清 0；在寄存器间接访问模式下，则把目的操作数所在的存贮空间的值所对应的地址存贮空间清 0。

COM 取反

COM dst

操 作： dst ← NOT dst
对目的操作数取反，并存入目的操作数寄存器。即如果相应位为 1，则操作后该位为 0；反之，依然。

标志位： C：不受影响。
Z：如果运算结果为 0,该位置 1；反之，置 0。
S：如果结果的第 7 位为 1，则该位为 1；反之，为 0。
V：该操作之后，此位为 0。

格 式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	60	R
		4	61	IR

编程实例： 假如 R1=07H，寄存器 07H=0F1H。

```
COM    R1       ；R1=0F8H
COM    @R1      ；R1=07H，07H=0EH
```

在第一个例子中，目的操作数寄存器 R1 中的值为 07H。COM 操作之后，所有逻辑 1 取反变为逻辑 0。结果为 R1=0F8H。在第二个例子中，间接寄存器访问模式下，把寄存器 07H 单元中的值取反。结果为 07H=0EH。

CP 比较

CP dst,src

操 作： dst ← src
源操作数与目的操作数进行比较，比较结果影响相应的标志位。源操作数和目的操作数都不变。

标志位： C：如果有借位时，该位置 1；反之，清 0。即（src>dst）
Z：如果结果为 0，则置 1；反之，清 0。
S：如果结果为负，则置 1；反之，清 0。
V：如果运算中产生溢出，则此位置 1；反之，清 0。

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst|src	2	4	A2	r	r
		6	A3	r	Ir
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

编程实例：1 假如 R1=02H，R2=03H

```
CP      R1，R2          ；C，S 标志位将会被置 1
```

该指令实际操作是 R1 减去 R2，由于 R1 小于 R2，C 被置 1，S 也被置 1。

2 假如 R1=05H，R2=0AH

```
      CP      R1，R2          ；比较 R1，R2 的值，
      JP      UGE，SKIP       ；R1 大于等于 R2 时跳转
      INC     R1              ；R1 加 1
SKIP：LD      R3，R1
```

DEC　　减 1

DEC　　dst

操　作：　dst ← dst – 1

目的操作数减 1，放入目的操作数寄存器。

标志位：　C：不受影响

Z：如果结果为 0，则置 1；反之，清 0。

S：如果结果为负，则置 1；反之，清 0。

V：如果算术运算产生溢出，则此位置 1。也即是说目的操作数的值为-128（80H），结果为+127（7FH）。反之，清 0。

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	00	R
		6	01	IR

编程实例：　假如 R1=03H，寄存器 03H=10H

```
DEC     R1      ;R1=02H
DEC     @R1     ;03H=0FH
```

在第一个例子中，R1 的值减 1 之后，值为 02H；在第二个例子中，则是把寄存器中的值作为地址，并把此地址单元中的值减 1。

DI　　禁止中断

DI

操　作：　SYM(2)← 0

把 SYM.2 清 0，禁止所有中断。但各相应中断仍置起中断标志位，只是 CPU 并不响应中断。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	8F

编程实例：　假如 SYM=04H

```
DI
```

如果 SYM 的值为 04H，执行此条指令后，SYM 的值为 00H，禁止所有中断。

EI　　允许中断

EI

操　作：　SYM(2) ← 1

EI 指令置 SYM.2 为 1，打开中断。如果这时相应的中断位置 1，则 CPU 开始响应服务程序。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	9F

编程实例：　假如 SYM=00H

EI

如果 SYM=00H，即禁止所有中断。执行 EI 指令，则打开所有中断。

IDLE　　空闲模式

IDLE

操　作：　IDLE 指令将停止 CPU 时钟，但允许系统时钟继续工作。当 CPU 处于 IDLE 模式下时，外部中断或内部中断都可以把 CPU 从空闲模式唤醒。这时，要允许（IRQ）。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	6F

编程实例：　IDLE

NOP

NOP

NOP

执行该指令，将停止 CPU 时钟，但系统时钟仍然工作。

INC　　加 1

INC　　dst

操　作：　dst ← dst + 1
目的操作数加 1，存在目的操作数存贮器。

标志位：C：不受影响
Z：如果结果为 0，则置 1；反之，清 0。
S：如果结果为负，则置 1；反之，清 0。
V：如果算术运算产生溢出，则此位置 1。也即是说目的操作数的值为-128（80H），结果为+127（7FH）。反之，清 0。

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	1	4	rE r=0 到 F	r
opc | dst	2	4	20	R
		4	21	IR

编程实例：　假如 R0=1BH，寄存器 00H=0CH，寄存器 1BH=0FH：

```
INC    R0     ；R0=1CH
INC    00H    ；00H=0DH
INC    @R0    ；R0=1BH，寄存器 1BH=10H
```

在第一个例子中，采用的是寄存器访问方法，R0 中的值被加 1。
在第二个例子中，采用的是寄存器访问方法，00H 中的值被加 1。
在第三个例子中，采用的是寄存器间接访问方法，故 1BH 中的值被加 1。

IRET　　中断返回

IRET　　IRET

操　作：　FLAGS← @SP
SP ← SP + 1
PC ← @SP
SP ← SP + 2
SYM.2 ← 1
该指令用于中断返回。当中断返回时，弹出 FLAGS，PC 值，并再次允许中断。

标志位：　所有标志位的值变回没产生中断时的状态。

格　式：

	字节数	机器周期	指令代码 (16 进制)
opc	1	10 12	BF

JP 跳转

JP cc,dst (条件代码)
JP dst

操 作： 如果 cc（条件代码）为真，则 PC ← dst
当条件转移代码为真时，则把要转移的地址压入 PC；反之，则执行 JP 后的指令。无条件转移指令则不需判断条件转移指令，直接跳转到目的地址。

标志位： 不受影响

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
cc|opc | dst	3	8	ccD cc=0 到 F	r
opc | dst	2	8	30	IRR

注释：

1 3 字节格式用于有条件跳转，2 字节格式用于无条件跳转。
2 在条件跳转指令的第一字节中，条件代码和操作指令都占 4 位。

编程实例： 假如 C=1，寄存器 00H=01H，寄存器 01H=20H

```
JP    C，LABLE_W     ；LABLE_W=1000H,PC=1000H
JP    @00H           ；PC=0120H
```

在第一个例子中，假设 C=1，则条件满足，把 LABLE_W 的地址压入 PC；如果条件不满足，则执行条件转移指令下面的一条指令。

在第二个例子中，没有条件转移代码，是不需要判断的，直接转移到目的地址。

JR　　相对跳转指令

JR　　cc,dst

操　作：　如果 cc 条件转移代码为真，PC ← PC + dst
如果具体的条件代码为真，执行该指令将会把偏移量加到当前 PC 中，执行当前 PC 计数器地址的指令。反之，则执行条件转移指令之后的指令代码。

标志位：　不受影响

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst|src	2	6	ccB cc=0 到 F	r

注释：在 2 字节指令格式的第一个字节中，条件代码和操作代码各占 4 位。

编程实例：　假如 C=1，LABLE_X=1FF7H

```
JR      C,LABLE_X       ; PC=1FF7H
```

如果 C=1，条件判断为真，则执行此指令后，程序跳转直 LABLE_X 处。反之，则条件转移指令后面的指令代码将会被执行。

LD　　数据传送

LD　　dst,src

操　作：　dst ← src

源操作数装入目的操作数所在的存贮空间。源操作数不变。

标志位：　不受影响

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
dst| opc | src	2	4	rC	r	IM
		4	r8	r	R
src | |opc	2	4	r9	R	r
			r=0 到 F		
opc | dst|src	2	4	C7	r	Ir
		4	D7	Ir	r
opc | src | dst	3	6	E4	R	R
		6	E5	R	IR
opc | dst | src	3	6	E6	R	IM
		6	D6	IR	IM
opc | src | dst	3	6	F5	IR	R
opc | dst|src | x	3	6	87	r	x[r]
opc | src|dst | x	3	6	97	x[r]	r

编程实例：假如 R0=01H，R1=0AH，寄存器 00H=01H，寄存器 01H=20H，
寄存器 02H=02H，LOOP=30H，寄存器 3A=0FFH

```
LD    R0,#10H          ; R0=10H
LD    R0,01H           ; R0=20H,寄存器 01H=20H
LD    01H,R0           ; 寄存器 01H=01H,R0=01H
LD    R1,@R0           ; R1=20H,R0=01H
LD    @R0,R1           ; R0=01H,R1=0AH,寄存器 01H=0AH
LD    00H,01H          ; 寄存器 00H=20H,寄存器 01H=20H
LD    02H,@00H         ; 寄存器 02H=20H,寄存器 00H=01H
LD    00H,#0AH         ;寄存器 00H=0AH
LD    @00H,#10H ; 寄存器 00H=01H,寄存器 01H=02H,寄存器 02H=02H
LD    R0,#LOOP[R1]     ;R0=0FFH,R1=0AH
LD    #LOOP[R0],R1     ;寄存器 31H=0AH,R0=01H,R1=0AH
```

LDC/LDE 传送数据

LDC/LDE dst,src

操 作： dst ← src

这条指令将从程序存贮空间或数据存贮空传输一个字节至目的存贮器。源操作数不受影响。LDC 用于程序存贮空间，LDE 用于数据存贮空间。编译器将会使 Irr,rr 在程序存贮空间偶数，数据存贮空间为奇数。

标志位： 不受影响

格 式：

	格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
1	opc | dst|src	2	10	C3	r	Irr
2	opc | src|dst	2	10	D3	Irr	r
3	opc | dst|src | XS	3	12	E7	r	XS[rr]
4	opc | src|dst | XS	3	12	F7	XS[rr]	r
5	opc | dst|src | XL_L | XL_H	4	14	A7	r	XL[rr]
6	opc | src|dst | XL_L | XL_H	4	14	B7	XL[rr]	r
7	opc | dst|0000 | DA_L | DA_H	4	14	A7	r	DA
8	Opc | src|0000 | DA_L | DA_H	4	14	B7	DA	r
9	opc | dst|0001 | DA_L | DA_H	4	14	A7	r	DA
10	Opc | src|0001 | DA_L | DA_H	4	14	B7	DA	r

注释：1　对于格式 5、6 的源操作数或者工作寄存器对不能用 0-1 工作寄存器对
2　对于格式 3、4 的目的地址 XS[rr]和源操作数地址 XS[rr]都是 1 个字节
3　对于格式 5、6 的目的地址 XL[rr]和源操作数地址 XL[rr]都是 2 个字节
4　对于格式 7、8 的 DA 和 r 的值是程序存贮器的，对于格式 9、10 的 DA 和 r 则用到的是数据存贮器的。

编程实例：　假如 R0=11H，R1=34H,R2=01H,R3=04H,R4=00H,R5=60H,程序存贮空间地址 0061H=AAH,0103H=4FH,0104H=1AH,0105H=6DH,1104H=88H,外部数据存贮器 0061H=BBH,0103H=5FH,0104H=2AH,0105H=7DH 1104H=98H.

```
LDC    R0,@RR2            ;R0=1AH,R2=01H,R3=04H,R0=(0104H)=1AH
LDE    R0,@RR2            ;R0=2AH,R2=01H,R3=04H
LDC    @RR2,R0            ;程序存贮空间 0104H=11H,R0,R2,R3 值不变。
LDE    @RR2,R0            ;数据存贮空间 0104H=11H, R0,R2,R3 值不变。
LDC    R0,#01H[RR4]       ;R0 的值为程序存贮空间 0061H 的值。
                          ;R0=AAH,R4=00H,R5=60H
LDE    R0,#01H[RR4]       ;R0 的值为程序存贮空间 0061H 的值。
                          ;R0=BBH,R4=00H,R5=60H
LDC    #01H[RR4],R0       ;11H 装入地址为 0061H（01H+0060H）的
                          ;程序存贮空间
LDE    #01H[RR4],R0       ;11H 装入地址为 0061H（01H+0060H）的
                          ;数据存贮空间
LDC    R0,#1000H[RR2]     ;程序存贮空间中,地址为 1104H(1000H+0104H)
                          ;的存贮单元中的值装入 R0 寄存器。
LDE    R0,#1000H[RR2]     ;数据存贮空间中;地址 1104H（1000H+0104H）
                          ;的存贮单元的值装入 R0 寄存器中。
LDC    R0,1104H           ;程序存贮空间 1104H 中的值装入 R0 寄存器。
                          ;源存贮空间的值不变。
LDE    R0,1104H           ;数据存贮空间 1104H 中的值装入 R0 寄存器。
                          ;源存贮空间的值不变。
LDC    1105H,R0           ;R0 寄存器中的值 11H 装入地址为 1105H 的程序
                          ;存贮空间,R0 寄存器中的值不变。
LDE    1105H,R0           ;R0 寄存器中的值 11H 装入地址为 1105H 的数据
                          ;存贮空间，R0 寄存器中的值不变。
```

注释：这些指令不支持掩模 ROM。
　　LDC 指令可用于程序查表。

LDCD/LDED 传送数据之后地址自动减 1

LDCD/LDED dst,src

操 作：

dst ← src
rr ← rr – 1
这条指令多用于对程序存贮区或数据存贮区堆栈进行数据操作或数据块操作。操作地址有指定的工作寄存器对提供。源操作数转入目的操作数存贮空间。操作之后，地址自动减 1。源操作数不受影响。LDCD 寻址程序存贮空间，LDED 寻址数据存贮空间。汇编器在汇编时,把访问程序存贮空间的”Irr”汇编为偶数，把访问数据存贮空间的“Irr”汇编为奇数。

标志位：

不受影响

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
opc | dst|src	2	10	E2	r	Irr

编程实例：

假如 R6=10H,R7=33H,R8=12H;程序存贮空间 1033H=0CDH,
外部数据存贮空间 1033H=0DDH

```
LDCD    R8,@RR6    ;程序存贮空间 1033H 单元中的值 0CDH 装入
                   ;寄存器 R8。
                   ;R8=0CDH,R6=10H,R7=32H(RR6←RR6-1)

LDED    R8,@RR6    ;数据存贮空间 1033H 单元中的值 0DDH 装入
                   ;寄存器 R8。
                   ;R8=0DDH,R6=10H,R7=32H(RR6←RR6-1)
```

LDCI/LDEI　传送数据之后自动加 1

LDCI/LDEI　dst ← src

rr ← rr + 1

这条指令多用于对程序存贮区或数据存贮区堆栈进行数据操作或数据块操作。操作地址有指定的工作寄存器对提供。源操作数转入目的操作数存贮空间。操作之后，地址自动加 1。源操作数不受影响。LDCD 寻址程序存贮空间，LDED 寻址数据存贮空间。汇编器在汇编时，把访问程序存贮空间的”Irr”汇编为偶数，把访问数据存贮空间的“Irr”汇编为奇数。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
opc | dst|src	2	10	E3	r	Irr

编程实例：　假如　R6=10H,R7=33H,R8=12H;程序存贮空间 1033H=0CDH,

外部数据存贮空间 1033H=0DDH

```
LDCI    R8,@RR6    ;程序存贮空间 1033H 单元中的值 0CDH 装入
                   ;寄存器 R8。
                   ;R8=0CDH,R6=10H,R7=34H(RR6←RR6+1)
LDEI    R8,@RR6    ;数据存贮空间 1033H 单元中的值 0DDH 装入
                   ;寄存器 R8。
                   ;R8=0DDH,R6=10H,R7=34H(RR6←RR6+1)
```

NOP　空操作（No Operqtion）

NOP

操　作：　CPU 执行此指令时,不作任何操作。一个或多个 NOP 可以起到延时作用。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	FF

编程实例：　NOP

此指令不作任何操作，只是延时一个执行周期。

OR　　逻辑或（Logical OR）

OR　　dst,src

操　作：　dst ← dst OR src

源操作数与目的操作数进行逻辑或运算，运算结果存放在目的操作数地址空间。源操作数不受影响。

标志位： C：不受影响

Z：如果结果为 0，则置 1；反之，清 0。

S：如果结果第 7 位为 1，则置 1；反之，清 0。

V：该操作之后，此位为 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst|src	2	4	42	r	r
		6	43	r	Ir
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

编程实例：　假如　R0=15H,R1=2AH,R2=01H,寄存器 00H=08H,寄存器 01H=37H,寄存器 08H=8AH

```
OR    R0,R1        ;R0=3FH,R1=2AH
OR    R0,@R2       ;R0=37H,R2=01H,寄存器 01H=37H
OR    00H,01H      ;寄存器 00H=3FH,寄存器 01H=37H
OR    01H,@00H     ;寄存器 00H=08H,寄存器 01H=0BFH
OR    00H,#02H     ;寄存器 00H=0AH
```

在第一个例子中，R0=15H，R1=2AH，则进行或操作的结果是 3FH，结果装入 R0。其余的例子，则是介绍了不同地址访问模式下的或操作。

POP 弹栈（Pop From Stack）

POP dst

操 作： dst ← @sp

sp ← sp + 1

堆栈指针所指的地址单元中的内容装入目的地址。堆栈指针加 1。

标志位： 不受影响

格 式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst|src	2	8	50	R
		8	51	IR

编程实例： 假如：寄存器 00H=01H，寄存器 01H=1BH，SP（0D9H）=0BBH，堆栈区寄存器 0BBH=55H

POP 00H ;寄存器 00H=55H，SP=0BCH

POP @00H ;寄存器 00H=01H,寄存器 01H=55H,SP=0BCH

在第一个例子中,寄存器 00H=01H,执行指令“POP 00H”之后,寄存器 00H=55H,同时，堆栈指针加 1。SP=0BCH。

PUSH 压栈(Push To Stack)

PUSH src

操 作： SP ← SP – 1

@SP ← src

进行压栈操作时，堆栈指针先减 1，而后，把源操作数压入堆栈。该操作会改变堆栈指针的值。

标志位： 不受影响

格 式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst|src	2	8	70	R
		8	71	IR

编程实例： 假如：寄存器 40H=4FH，寄存器 4FH=0AAH，SP=0C0H

PUSH 40H ;寄存器 40H=4FH,0BFH=4FH

PUSH @40H ;寄存器 40H=4FH,4FH=0AAH,0BFH=0AAH,SP=0BFH

在第一个例子中，寄存器 40H=4FH，执行指令之后，堆栈指针减 1，把 40H 中的值压入堆栈，故 0BFH=4FH，SP=0BFH。

RCF　　C 清 0（Reset Carry Flag）

RCF

操　作：　C ← 0

该指令将 C 清 0

标志位：C：清 0

其余不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	CF

编程实例：　假如：C=‘1’或‘0’

RCF

该指令把 C 清为逻辑 0。

RET　　子程序返回（Return）

RET

操　作：　PC ← @SP

SP ← SP + 2

RET 指令经常用于子程序返回。执行此指令，将把压入堆栈中的ＰＣ值弹到 PC 中。CPU 执行新的指令。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	8 10	AF

编程实例：　假如：SP=0BCH,(SP)=101AH,PC=1234

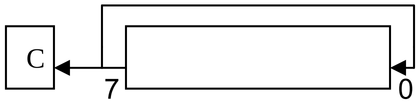
RET　　　　;PC=101AH,SP=0BEH

RET 指令把 0BCH 单元中的值弹到 PC 高字节，把 0BDH 单元中的值弹到 PC 低字节。开始执行 101AH 单元的指令。堆栈指针 SP==0BEH。

RL 左移（Rotate Left）

RL dst

操 作： C ← dst(7)
dst(0) ← dst(7)
dst(n+1)← dst(n),n=0-6
目的操作数左移一个位置。原第 7 位数移到最低位，同时，也移到 C。



标志位： C：如果第 7 位为 1，在执行此指令之后，此位置 1。
Z：如果结果为 0，则置 1；反之，清 0。
S：如果操作结果的第 7 位为 1，则置 1；反之，清 0。
V：如果有符号数在操作中改变，即产生算术溢出，则置 1；反之，清 0。

格 式：

opc	dst|src	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
		2	4	90	R
			4	91	IR

编程实例： 假如：寄存器 00H=0AAH，寄存器 01H=02H，寄存器 02H=17H
RL 00H ；寄存器 00H=55H，C='1'
RL @01H ；寄存器 01H=02H，寄存器 02H=2EH，C='0'

在第一个例子中，寄存器 00H=0AAH（10101010B），执行左移指令之后，00H 中的值为 55H（01010101B）。同时，置 C 和 V。

RLC　　带进位左移（Rotate Left Carry）

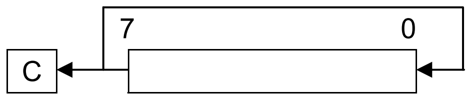
RLC　　dst

操　作：　dst(0) ← C

C ← dst(c)

dst(n+1) ← dst(n),n=0-6

目的操作数和 C 一起左移一位。原操作数的第 7 位，移入进（借）位。进（借）位移入第 1 位



标志位： C：如果第 7 位为 1，在执行此指令之后，此位置 1。

Z：如果结果为 0，则置 1；反之，清 0。

S：如果操作结果的第 7 位为 1，则置 1；反之，清 0。

V：如果有符号数在操作中改变，即产生算术溢出，则置 1；反之，清 0。

格　式：

opc	dst	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
		2	4	10	R
			4	11	IR

编程实例：　假如：寄存器 00H=0AAH，寄存器 01H=02H，寄存器 02H=17H，C=‘0’

```
RLC    00H     ;寄存器 00H=54H，C=‘1’
RLC    @01H    ;寄存器 01H=02H，寄存器 02H=2EH，C=‘0’
```

在第一个例子中，寄存器 00H=0AAH（10101010B），执行此指令，00H=55H(01010101B)原操作数的第 7 位把 C 置 1，同时，也置 V 为 1。

RR　　右移（Rotate Right）

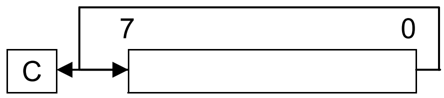
RR　　dst

操　作：　C ← dst(0)

dst(7） ← dst(0)

dst(n) ← dst(n + 1)，n=0-6

目的操作数向右移一位。操作数的最低位移到最高位（7），同时，也移到进（借）位。



标志位： C：如果第 0 位为 1，在执行此指令之后，此位置 1。

Z：如果结果为 0，则置 1；反之，清 0。

S：如果操作结果的第 7 位为 1，则置 1；反之，清 0。

V：如果有符号数在操作中改变，即产生算术溢出，则置 1；反之，清 0。

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	E0	R
		4	E1	IR

编程实例：　假如：寄存器 00H=31H，寄存器 01H=02H，寄存器 02H=17H

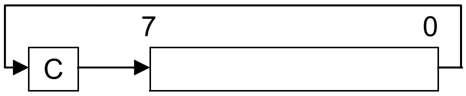
RR	00H	;寄存器 00H=98H，C=‘1’
RR	@01H	;寄存器 01H=02H，寄存器 02H=8BH，C=‘1’

在第一个例子中，寄存器 00H=31H，右移将 00H 单元中各位右移一位。原操作数的第 0 位移到第 7 位。右移之后，寄存器 00H=98H。在右移的过程中，第 0 位也移到了进（借）位中。溢出位（V）也置 1。

RRC 带进位右移（Rotate Right Through Carry）

RRC dst

操　作： dst(7) ← C
C ← dst(0)
dst(n) ← dst(n + 1),n=0-6
目的操作数和进（借）位一起循环右移。原操作数的第 0 位移入进（借）位，
进（借）位则移到第 7 位。



标志位： C：如果第 0 位为 1，在执行此指令之后，此位置 1。
Z：如果结果为 0，则置 1；反之，清 0。
S：如果操作结果的第 7 位为 1，则置 1；反之，清 0。
V：如果有符号数在操作中改变，即产生算术溢出，则置 1；反之，清 0。

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	C0	R
		4	C1	IR

编程实例： 假如：寄存器 00H=55H，寄存器 01H=02H，寄存器 02H=17H，C='0'

```
RRC    00H     ;寄存器 00H=2AH，C='1'
RRC    @00H    ;寄存器 01H=02H，寄存器 02H=0BH,C='1'
```

在第一个例子中，寄存器 00H=55H，带进位右移把 00H 单元的值和 C 一起循环右移。原操作数的第 0 位移到 C，C 移到第 7 位。操作之后，寄存器 00H=2AH。标志位 S=0，V=0。

SBC 带进位减法（Subtract With Carry）

SBC dst,src

操 作： dst ← dst –src – C

目的操作数减去原操作数再减去当前 C，所得结果存放在目的操作数存贮器。源操作数不受影响。减法操作实际进行的是补码的加法。在多字节减法中，为了进行精确的减法操作，需用到带进位减法指令。

标志位： C：如果产生借位，此位置 1；反之，清 0。

Z：如果结果为 0，则置 1；反之，清 0。

S：如果操作结果为负，则置 1；反之，清 0。

V：如果两个操作数的符号相反，但操作结果的符号与源操作数的符号相同，则产生算术溢出，置此位为 1；反之，清 0。

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	src
opc | dst|src	2	4	32	r	r
		6	33	r	Ir
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	M

编程实例： 假如：R1=10H，R2=03H，C=‘1’，寄存器 01H=20H,寄存器 02H=03H，寄存器 03H=0AH

```
SBC    R1,R2       ;R1=0CH,R2=03H
SBC    R1,@R2      ;R1=05H,R2=03H,寄存器 03H=0AH
SBC    01H,02H     ;寄存器 01H=1CH,寄存器 02H=03H
SBC    01H,@02H ;寄存器 01H=15H,寄存器 02H=03H,寄存器 03H=0AH
SBC    01H,#8AH    ;寄存器 01H=95H,C,S,V=1
```

在第一个例子中，R1=10H，R2=03H，执行带进位减法，则（R1）-（R2）-C=0CH，R1=0CH，原操作数不受影响。

SCF　　C 置 1（Set Carry Flag）

SCF

操　作：　C ← 1

进（借）位被置为逻辑 1。

标志位：　C：置为逻辑 1。

其它标志位不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）
opc	1	4	DF

编程实例：　SCF

置 C 为 1

STOP　　冻结状态

STOP

操　作：　STOP 指令冻结 CPU 时钟和系统时钟，使微控制器进入冻结模式。在冻结模式，片内寄存器，控制寄存器，I/O 口数据寄存器的值不会丢失。复位操作或外部中断可使 CPU 退出冻结状态。对于外部复位操作，必须要有足够时间的低电平，以保证使晶振稳定。

标志位：　不受影响

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc	1	4	7F	—	—

编程实例：

```
LD      STOPCON,#0A5H
STOP
NOP
NOP
NOP
```

进入冻结状态，CPU 停止一切操作。当 STOPCON 寄存器的值不是 0A5H 时，STOP 指令将会使 PC=0100H。

SRA　　算术右移（Shift Right Arithmetic）

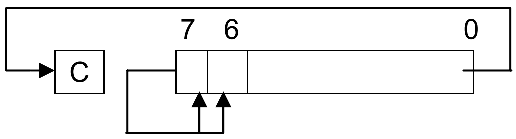
SRA　　dst

操　作：　dst(7) ← dst(7)

C ← dst(0)

dst(n) ← dst(n + 1),n=0-6

算术右移指令中，操作数第 0 位移到进（借）位，第 7 位（符号位）不变，同时，第 7 位的值再移到第 6 位。



标志位： C：如果第 0 位为 1，在执行此指令之后，此位置 1。

Z：如果结果为 0，则置 1；反之，清 0。

S：如果结果为负，则置 1；反之，清 0。

V：清为 0。

格　式：

	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst
opc | dst	2	4	D0	R
		4	D1	IR

编程实例：　假如：寄存器 00H=9AH，寄存器 02H=03H，寄存器 03H=0BCH，C=‘1’

```
SRA    00H     ;寄存器 00H=0CDH，C=‘0’
SRA    @02H    ;寄存器 02H=03H，寄存器 03H=0DEH，C=‘0’
```

在第一个例子中，寄存器 00H=9AH，算术右移指令把第 0 位移到 C，第 7 位移到第 6 位，第 7 位保持不变。操作之后，寄存器的值 00H=0CDH。

SUB 减法（Subtract）

SUB dst,src

操　作： dst ← dst – src

目的操作数减去源操作数，结果存放在目的操作数寄存器。源操作数不变。减法操作实际上进行的操作是，被减数加上减数的补码。

标志位： C：如果产生借位，则此位置 1；反之，清 0。

Z：如果结果为 0，则置 1；反之，清 0。

S：如果结果为负，则置 1；反之，清 0。

V：如果两个操作数的符号相反，但操作结果的符号与源操作数的符号相同，则产生算术溢出，置此位为 1；反之，清 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst	2	4	22	r	r
		6	23	r	Ir
opc | src | dst	3	6	24	R	R
		6	25	R	IR
opc | dst | src	3	6	26	R	IM

编程实例： 假如：R1=12H，R2=03H，寄存器 01H=21H，寄存器 02H=03H，寄存器 03H=0AH

```
SUB    R1,R2          ;R1=0FH,R2=03H
SUB    R1,@R2         ;R1=08H,R2=03H
SUB    01H,02H        ;01H=1EH,02H=03H
SUB    01H,@02H       ;寄存器 01H=17H,寄存器 02H=03H
SUB    01H,#90H       ;寄存器 01H=91H,C,S,V='1'
SUB    01H,#65H       ;寄存器 01H=0BCH,C,S='1',V='0'
```

在第一个例子中，R1=12H，R2=03H，减法操作（R1）-（R2）=0FH。操作之后，R1=0FH。

TCM　　取反测试（Test Complement Under Mask）

TCM　　dst,src

操　作：　(~dst) & src

这条指令测试目的操作数指定位是否为逻辑 1。可以把源操作数的相应位设置为 1，以便进行测试。TCM 指令先把目的操作数取反，而后再与源操作数进行与运算。可以通过标志位检测操作结果。目的操作数和源操作数都不受影响。

标志位：　C：不受影响

Z：如果结果为 0，则置 1；反之，清 0。

S：如果结果第 7 位为 1，则置 1；反之，清 0。

V：清为 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst	2	4	72	r	r
		6	73	r	Ir
opc | src | dst	3	6	74	R	R
		6	75	R	IR
opc | dst | src	3	6	76	R	IM

编程实例：　假如：R0=0C7H,R1=02H,R2=12H,寄存器 00H=2BH，寄存器 01H=02H，寄存器 02H=23H

```
TCM    R0,R1       ;R0=0C7H,R1=02H,Z='1'
TCM    R0,@R1      ;R0=0C7H,R1=02H,02H=23H,Z='0'
TCM    00H,01H     ;寄存器 00H=2BH,寄存器 01H=02H,Z='1'
TCM    00H,@01H    ;寄存器 00H=2BH,寄存器 01H=02H
TCM    00H,#34H    ;寄存器 00H=2BH,Z='0'
```

在第一个例子中，R0=0C7H，R1=02H，"TCM　R0,R1"测试目的操作数的第 1 位是否为 1，该操作把目的操作数取反，而后与源操作数进行与运算。操作结果，Z=1。

TM　　测试（Test Under Mask）

TM　　dst,src

操　作：　dst & src

该指令测试目的操作数指定位是否为 0。可把原操作数的相应位置 1，以便测试。TM 指令把源操作数和目的操作数进行与运算。可以通过检测标志位（Z）检测操作结果。源操作数和目的操作数不受影响。

标志位：
C：不受影响
Z：如果结果为 0，则置 1；反之，清 0。
S：如果结果第 7 位为 1，则置 1；反之，清 0。
V：清 0。

格　式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst|src	2	4	72	r	r
		6	73	r	Ir
opc | src | dst	3	6	74	R	R
		6	75	R	IR
opc | dst | src	3	6	76	R	IM

编程实例：　假如: R0=0C7H,R1=02H,R2=18H,寄存器 00H=2BH，寄存器 01H=02H，寄存器 02H=23H

```
TM    R0,R1        ;R0=0C7H,R1=02H,Z="0"
TM    R0,@R1       ;R0=0C7H,R1=02H,寄存器 02H=23H,Z="0"
TM    00H,01H      ;寄存器 00H=2BH,寄存器 01H=02H,Z="0"
TM    00H,@01H     ;寄存器 02H=23H,Z="0"
TM    00H,#54H     ;寄存器 00H=2BH,Z="1"
```

在第一个例子中，R0=0C7H，R1=02H。"TM　R0,R1"测试目的操作数的第 1 位是否为 0，该操作把目的操与源操作数进行与运算。操作结果，Z=0。

XOR 异或（Logical Exclusive OR）

XOR dst,src

操 作： dst ← dst ⊕ src

源操作数与目的操作数进行异或操作，结果存在目的操作数存贮器。异或操作是，源操作数与目的操作数相应位相同，则为 0，不同则为 1。

标志位: C：不受影响
Z：如果结果为 0，则置 1；反之，清 0。
S：如果结果第 7 位为 1，则置 1；反之，清 0。
V：清 0。

格 式：

格式	字节数	机器周期	指令代码（16 进制）	地址访问模式 dst	地址访问模式 src
opc | dst|src	2	4	B2	r	r
		6	B3	r	Ir
opc | src | dst	3	6	B4	R	R
		6	B5	R	IR
opc | dst | src	3	6	B6	R	IM

编程实例： 假如 :R0=0C7H，R1=02H，R2=18H，寄存器 00H=2BH，寄存器 01H=02H，寄存器 02H=23H

```
XOR    R0,R1        ;R0=0C5H,R1=02H
XOR    R0,@R1       ;R0=0E4H,R1=02H,寄存器 02H=23H
XOR    00H,01H      ;寄存器 00H=29H,寄存器 01H=02H
XOR    00H,@01H     ;寄存器 00H=08H,寄存器 01H=02H,
                     寄存器 02H=23H
XOR    00H,#54H     ;寄存器 00H=7FH
```

在第一个例子中，R0=0C7H,R1=02H.异或操作的结果是 R0=0C5H。
其余几个例子是不同的地址访问模式下的异或操作。

第七章 时钟电路

概述

通过 Smart Option (3FH.1-0)的设置，可以选择内部 RC 震荡或外部晶振震荡。如果用内部晶振，则 X_{IN}(P1.0),X_{OUT}(P1.1)可以用作普通 I/O 口。内部晶振提供的时钟震荡频率为 3.2M 或 0.5M(VDD=5V)，选择何种震荡频率，也可通过 Smart Option 位来选择。

外部 RC 震荡对 S3C9444/C9454 提供 4M 时钟频率。S3C9444/C9454 内部集成有电容以满足震荡。外部石英晶振或陶瓷晶振提供 10M 时钟频率。S3C9444/C9454 的时钟外部连接如下：

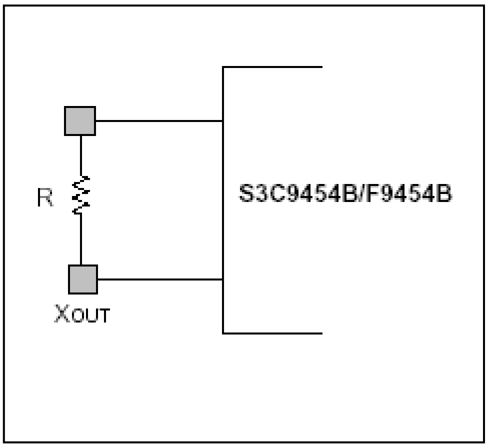


图 7-1 主晶振电路
（内部 RC 震荡）

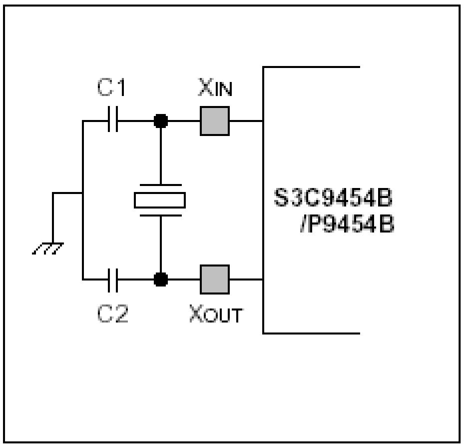


图 7-2 主晶振电路
（石英/陶瓷晶振）

为降低电流消耗，P1.0,P1.1 应设置为输出模式。

为了增加处理速度，减少噪声，在主晶振电路中没有加入分频电路。在实际操作中，需要整形电路，把波形信号整形为方波信号，以便于 CPU 处理逻辑操作。

掉电模式下时钟电路的状态

S3C9444/9454 有两种掉电模式，冻结状态或空闲模式。

---在冻结模式下，主晶振冻结，CPU 停止操作，内部中断及定时器停止工作。寄存器卷中的值和当前特殊功能寄存器的值都保持原数值。在复位操作或外部中断的情况下，CPU 退出冻结模式，晶振起振。

---在空闲模式下，主晶振关掉 CPU 的时钟，但内部中断及定时器仍然工作。当前 CPU 的工作状态被保存，包括堆栈指针，PC 值，系统状态寄存器。寄存器卷中的数值仍然保存。复位或中断（内部中断或外部中断）可以使 CPU 退出空闲模式。

系统时钟控制寄存器

系统时钟寄存器的地址是 D4H，它是可以读写访问的，有如下功能：

--允许/禁止晶振 IRQ 唤醒（CLKCON.7）

--系统时钟分频 1，2，8，16（CLKCON.4,CLKCON.3）

CLKCON 控制着是否允许外部中断唤醒 CPU，退出冻结模式（STOP）以及系统时钟分频。复位操作之后，允许 IRQ。同时，选择 f=fosc/16 作为 CPU 时钟。编程者可以选择分频时钟频率。

系统时钟控制寄存器

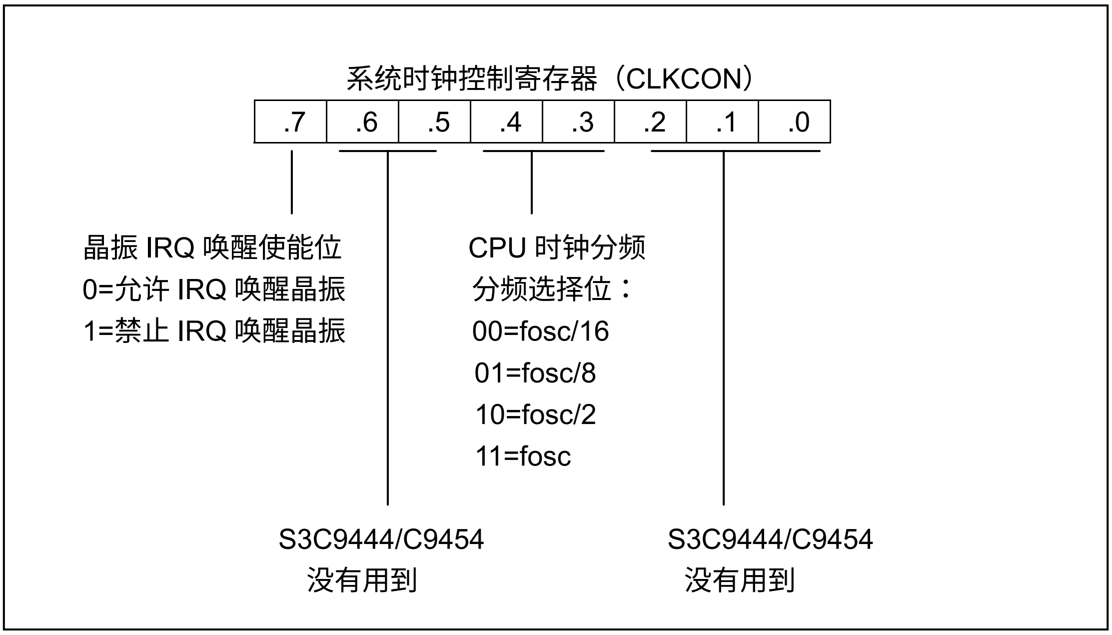


图 7-3　　系统时钟控制寄存器（CLKCON）

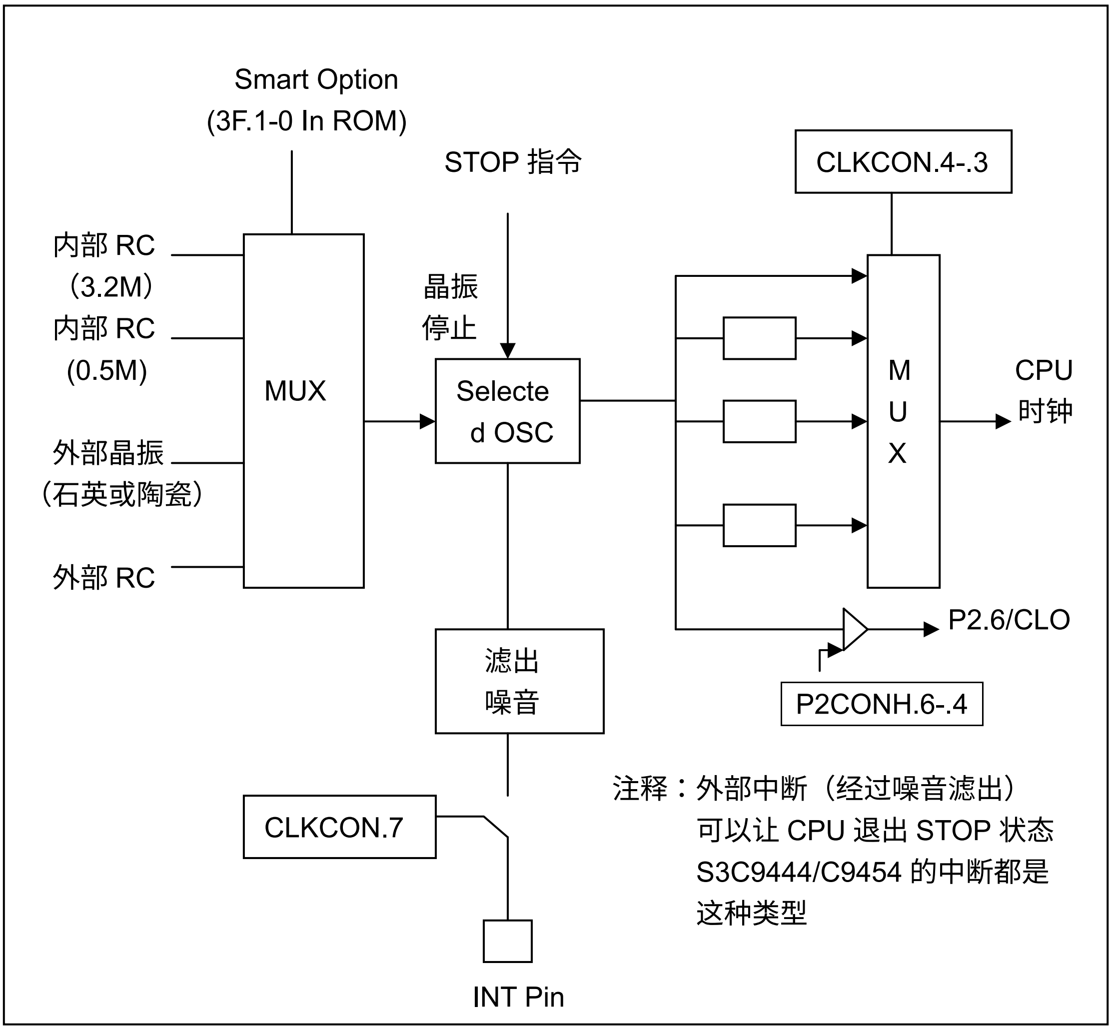


图 7-4　　系统时钟电路原理图

第八章　　　复位和掉电模式

复位电路

概述

通过设置 Smart Option（3E.7 in ROM）可以选择内部低电压复位（LVR）或外部复位。如果用内部低电压复位（LVR），则 P1.2（复位引脚）可以用作普通 I/O 口。

S3C9444/9454 可以有 4 种复位方法：

--通过外部上电复位。

--通过拉低复位引脚进行复位

--通过 WACTHDOG Timer 的溢出复位

--通过内部的低电压（LVR）复位

如果采用外部上电复位，当 VDD 引脚为高电平时，RESET 引脚为低电平，复位信号通过施密特触发电路与 CPU 时钟同步，从而复位系统。为了保证 CPU 时钟晶振稳定，复位信号一定要保持低电平一定时间。S3C9444/C9454 时钟稳定的最小时间为
6.55ms(≈2^{16}/fosc
fosc=10M)。

在正常操作模式下，VDD 及 RESET 均为高电平。当 RESET 引脚为低电平时，系统将会复位。复位后，各寄存器恢复为默认值。

S3C9444/C9454 带有看门狗功能，以防程序跑飞。如果程序没有在额定时间内清除看门狗计数器，看门狗计满溢出后，就会复位 MCU。

片内低电压复位是这样的：当电压低于设定电压时，系统处于复位状态；当电压大于设定电压时，系统开始工作。

为了保证当系统退出 STOP 状态时，有一定的时间稳定晶振，在进入 STOP 状态时，要设置好 BTCON。如果不希望用看门狗功能，你可以把'1010B'写到 BTCON 的高 4 位。

MCU 初始化顺序

下面是复位操作之后，MCU 的初始化顺序：

--禁止所有中断

--允许看门狗功能

--P0-P2 口设为输入模式

--禁止控制寄存器和数据寄存器并设为默认值

--PC 中装入 0100H（程序开始地址）

--当晶振稳定后，调入程序存贮空间 0100H 单元的指令并执行。

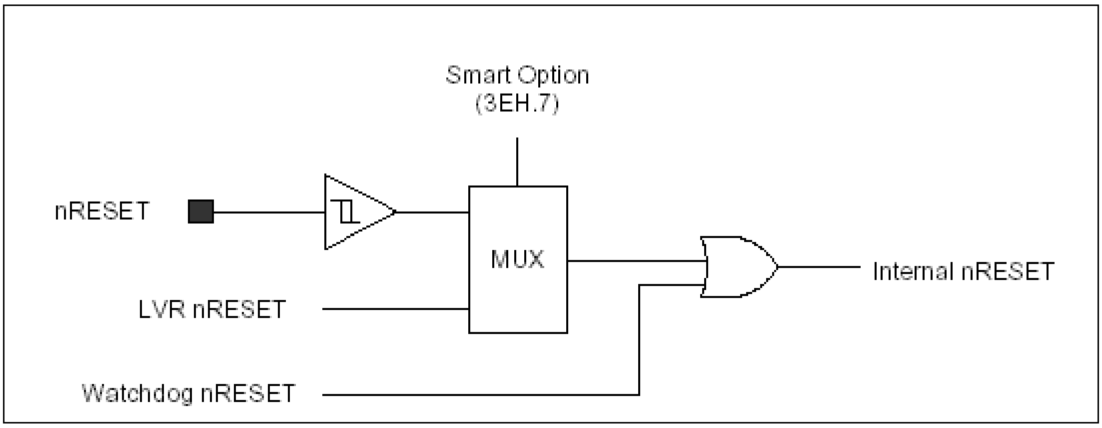


图 8-1　复位电路原理图

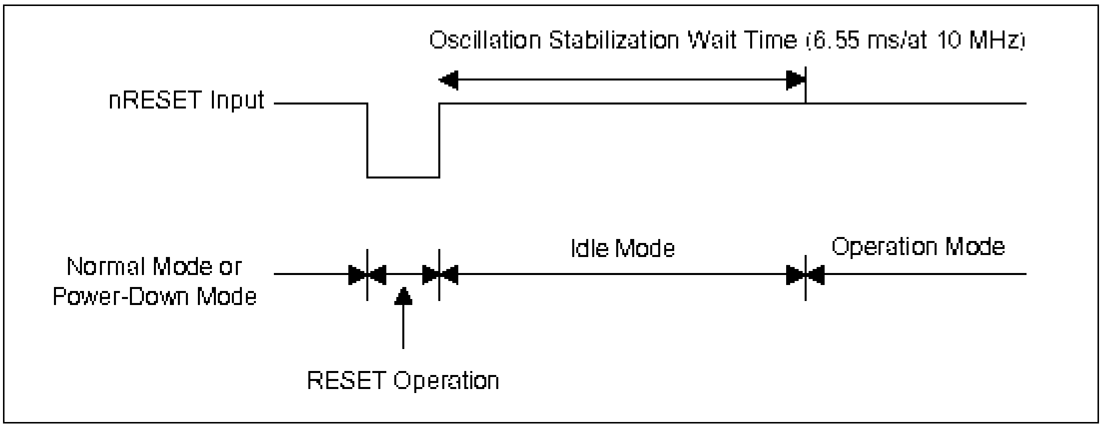


图 8-2　复位后 S3C9454B 的延时原理图

掉电模式

STOP 状态

STOP 指令将会使系统进入 STOP 状态，即冻结模式。在冻结模式下，主晶振冻结，CPU 停止操作，内部中断及定时器停止工作。寄存器卷中的值和当前特殊功能寄存器的值都保持原数值。电流消耗小于 100μA。复位操作或外部中断可以退出 STOP 状态。

复位操作退出 STOP 状态

当复位信号变为高电平时，系统退出 STOP 状态。控制寄存器恢复为默认值，所有寄存器的值保持没有复位以前的值。CPU 时钟选择 f=fosc/16，待时钟稳定后，从 0100H 单元取指令开始执行。

利用外部中断退出 STOP 状态

外部中断可以让系统退出 STOP 状态。内部中断，如果用外部时钟，也可以让系统退出 STOP 状态，如定时器、SIO 等。S3C9444/C9454 的 INT0、INT1 可以使系统退出 STOP 状态。

利用外部中断退出 STOP 状态，则控制寄存器和系统寄存器卷中的值不变，CPU 的处理速度不变，即 CPU 的时钟分频选择不变。利用外部中断退出 STOP 状态，在进入 STOP 状态时，要设置 BTCON。这样，在退出 STOP 状态时，有时间稳定晶振。

在退出 STOP 状态时，系统处理中断服务程序。中断处理完毕返回后，执行 STOP 指令后面的指令。

IDLE 状态

IDLE 指令将会使系统进入 IDLE 状态，即空闲模式。在空闲模式下，主晶振关掉 CPU 的时钟，但内部中断及定时器仍然工作。当前 CPU 的工作状态被保存，包括堆栈指针，PC 值，系统状态寄存器。寄存器卷中的数值仍然保存。各 I/O 口仍然保持原来模式。

复位操作或中断可以使系统退出 IDLE 状态。

用复位操作退出 IDLE 模式后，系统状态同用复位操作退出 STOP 状态时一样。

不管外部中断或者内部中断，都可以使系统退出 IDLE 状态。

注释： 只有外部中断可以使系统退出 STOP 状态。不管外部中断或内部中断都可以使系统退出 IDLE 状态。
在进入 STOP 状态或 IDLE 状态之前，应关掉 ADC，否则，电流消耗会非常大。

控制寄存器复位值

下表列出了系统寄存器、控制寄存器复位后的值。

— ‘1’、‘0’代表复位后此位为逻辑 1 或逻辑 0。

— ‘x’代表复位后此位值不确定。

— ‘-’代表此位没有用到

表 8-1　　寄存器复位值

寄存器名	符　号	地址&位置		寄存器值							
		地址	R/W	7	6	5	4	3	2	1	0
定时器 0 计数寄存器	T0CNT	D0H	R/W	0	0	0	0	0	0	0	0
定时器 0 数据寄存器	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
定时器 0 控制寄存器	T0CON	D2H	R/W	0	0	-	-	0	-	0	0
D3H 没有用到											
时钟控制寄存器	CLKCON	D4H	R/W	0	-	-	0	0	-	-	-
系统标志寄存器	FLAGS	D5H	R/W	X	X	X	X	-	-	-	-
D6H-D8H 没有用到											
堆栈指针寄存器	SP	D9H	R/W	X	X	X	X	X	X	X	X
DAH 没有用到											
MDS 特殊寄存器	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
基本定时控制寄存器	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
基本定时计数器	BTCNT	DDH	R	0	0	0	0	0	0	0	0
测试模式寄存器	FTSTCON	DEH	W	-	-	0	0	0	0	0	0
系统模式寄存器	SYM	DFH	R/W	-	-	-	-	-	0	0	0

(续)

P0 数据寄存器	P0	E0H	R/W	0	0	0	0	0	0	0	0
P1 数据寄存器	P1	E1H	R/W	-	-	-	-	-	0	0	0
P2 数据寄存器	P2	E2H	R/W	-	0	0	0	0	0	0	0
E3H-E5H 没有用到											
P0 控制寄存器（高字节）	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
P0 控制寄存器	P0CON	E7H	R/W	0	0	0	0	0	0	0	0
P0 中断相应控制寄存器	P0PND	E8H	R/W	-	-	-	-	0	0	0	0
P1 控制寄存器	P1CON	E9H	R/W	0	0	-	-	0	0	0	0
P2 控制寄存器（高字节）	P2CONH	EAH	R/W	-	0	0	0	0	0	0	0
P2 控制寄存器（低字节）	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
ECH-F1H 没有用到											
PWM 数据寄存器	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM 控制寄存器	PWMCON	F3H	R/W	0	0	-	0	0	0	0	0
STOP 控制寄存器	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
F5H-F6H 没有用到											
A/D 控制寄存器	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D 转换数据寄存器 (高字节)	ADDATAH	F8H	R	X	X	X	X	X	X	X	X
A/D 转换数据寄存器（低字节）	ADDATAL	F9H	R	0	0	0	0	0	0	X	X
FAH-FFH 没有用到											

☞ 编程实例:

```
;-----------------<<中断向量地址>>
                    ORG         0000H
                    VECTOR    00H,INT_9454
;------------------<<Smart Option>>
                    ORG         003CH
                    DB          00H              ;必须初始化为 0
                    DB          00H              ;必须初始化为 0
                    DB          0E7H             ;允许低电压复位
                    DB          03H              ;内部 RC 震荡 3.2M
;-------------------<<初始化>>
                    ORG         0100H
RESET:              DI                           ;禁止中断
                    LD      BTCON,#10100011B       ;禁止看门狗
                    LD      CLKCON,#00011000B      ;选择 CPU 时钟频率
                    LD      SP,#0C0H
                    LD      P0CONH,#10101010B      ;
                    LD      P0CONL,#10101010B      ;P0.0—P0.7 为推挽式输出
                    LD      P1CON,#00001010B       ;P1.0-P1.1 为推挽式输出
                    LD      P2CONH,#01001010B      ;
                    LD      P2CONL,#10101010B      ;P2.0-P2.6 为推挽式输出
;--------------------<<T0  设置>>
                    LD      T0DATA,#50H
                    LD      T0CON,#01001010B
;--------------------<<清除寄存器单元 00H—5FH 单元的值>>
                    LD      R0,#0
RAM_CLR:            CLR      @R0
                    INC      R0
                    CP       R0,#0BFH
                    JP       ULE,RAM_CLR
;--------------------<<   >>
                    .
                    EI
;-------------------<<主函数>>
MAIN:               NOP
```

```
                LD          BTCON,#02H
                .
                .
                CALL        KEY_SCAN
                .
                .
                CALL        LED_DISPLAY
                .
                .
                .
                CALL        JOB
                .
                .
                .
                JR          T,MAIN
;------------------ <<子程序>>
KEY_SCAN:       NOP
                .
                .
                RET
LED_DISPLAY:    NOP
                .
                .
                .
                RET
JOB:            NOP
                .
                .
                RET
;-------------------<<中断服务程序>>
INT_9454:       TM          T0CON,#00000010B     ;是否允许 T0 中断
                JR          Z,NEXT_CHK1
                TM          T0CON,#00000001B     ;T0 是否产生中断
                JP          NZ,INT_TIMER0
```

```
NEXT_CHK1:  TM      PWMCOM,#00000010B   ;PWM 溢出中断使能检测
            JR      Z,NEXT_CHK2
            TM      P0PND,#00000001B
            JP      NZ,PWMOVF_INT
NEXT_CHK2:  TM      P0PND,#00000010B    ;INT1 中断使能检测
            JP      Z,NEXT_CHK3
            TM      P0PND,#00000001B
            JP      NZ,INT0_INT
NEXT_CHK3:  TM      P0PND,#00001000B
            JP      Z,END_INT
            TM      P0PND,#00000100B
            JP      NZ,INT1_INT
            IRET
END_INT:    IRET
;-----------------------<<T0 中断服务程序>>
INT_TIMER0:   .
            .
            AND     T0CON,#11110110B    ;清除中断标志位
            IRET
;------------------------<<PWM 溢出中断>>
PWMOVF_INT:   .
            .
            AND     PWMCON,#11110110B；清除中断标志位
            IRET
;-----------------------<<外部中断 0>>
INT0_INT:     .
              .
            AND     P0PND,#11111110B    ；清除中断标志位
            IRET
;-----------------------<<外部中断 1>>
INT1_INT:     .
              .
            AND     P0PND,#11111011B    ；清除中断标志位
            IRET
            .END
```

第九章　　I/O 口

概述

S3C9444/C9454 有三个 I/O 口，共 18 个引脚，可以直接访问这些数据寄存器。所有 I/O 口都可以直接驱动 LED（典型电流输出为 10mA）。

表 9-1　S3C9444/C9454I/O 口功能综述

I/O 口	功　能　描　述	可访问方法
0	可对该口的每一位进行功能设定，可以设定为施密特触发器输入或推挽式输出。P0 口也可用于相关功能设定，如 A D C，外部中断。	可按位进行访问
1	可对该口的每一位进行功能设定，可以设定为施密特触发器输入、推挽式输出或开漏输出。P1 口也可由 Smart Option 设定为晶振输入、复位输入。P1.2 只能设为输出。	可按位进行访问
2	可对该口的每一位进行功能设定，可以设定为施密特触发器输入、推挽式输出或开漏输出。P2 口也可设定为相关功能，如 ADC、CLO、T0 时钟输入	可按位进行访问

下表列出了各输出口的数据寄存器名、地址和读写特性。

表 9-2　个口数据寄存器综述

寄存器名	符　号	16 进制	R/W
P0 数据寄存器	P1	E0H	R/W
P1 数据寄存器	P2	E1H	R/W
P2 数据寄存器	P3	E2H	R/W

注释：复位之后，个口数据寄存器的值为 00H

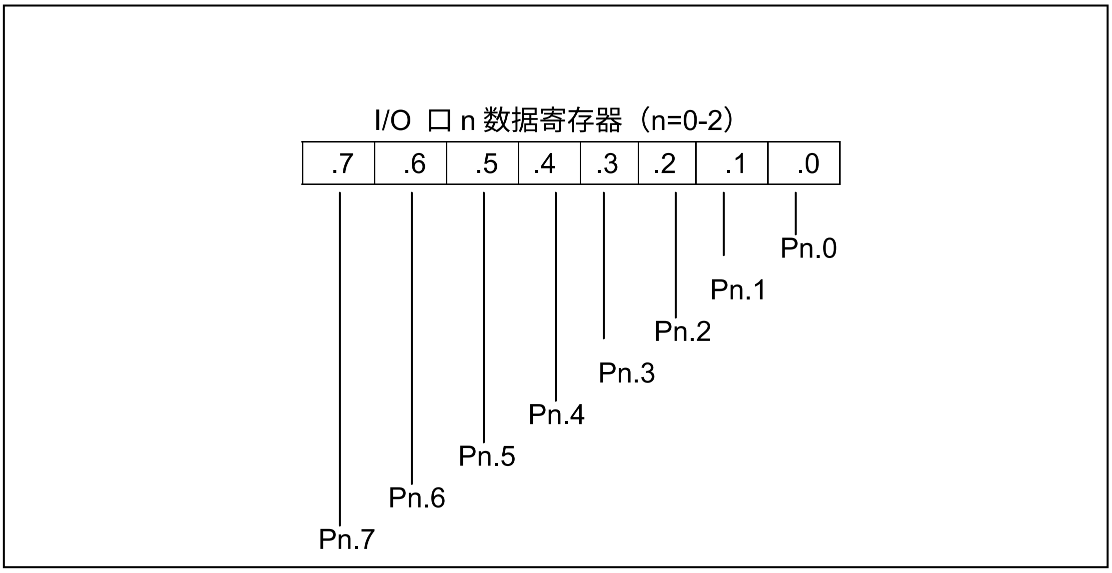


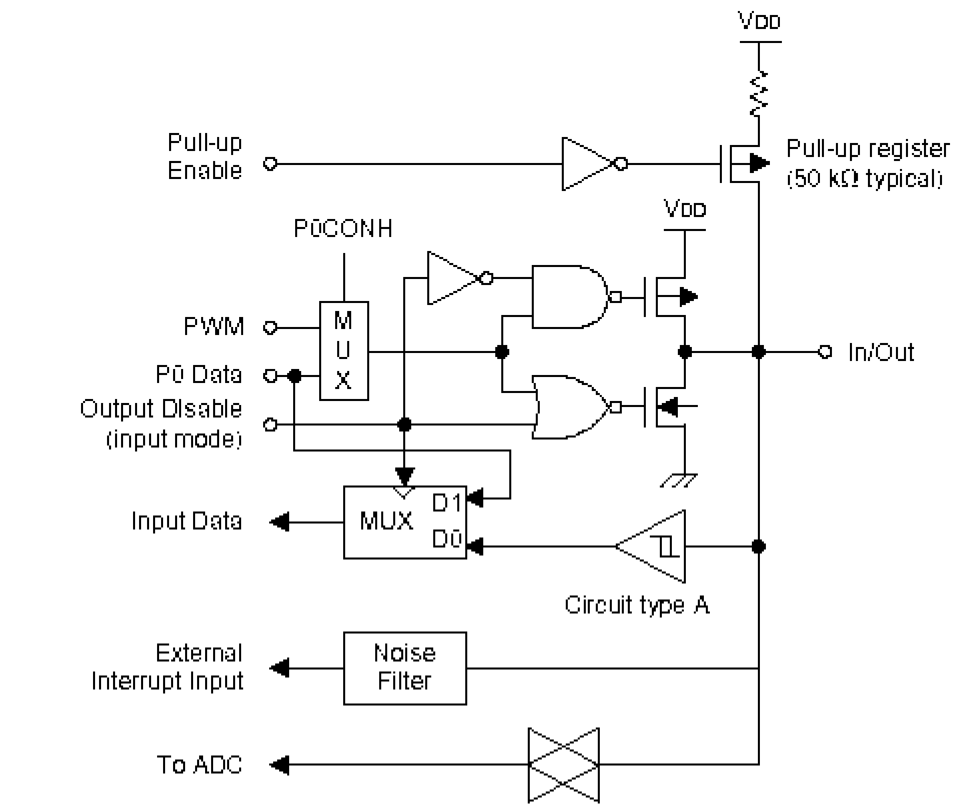
图 9-1　　各口数据寄存器格式

P0 口

P0 口是可位编程的通用型 I/O 口。你可以选择输入模式或推挽式输出模式。另外，你可以利用控制寄存器设置不同引脚的上拉电阻。该口的输出电流较大，可以直接驱动 LED（电流约为 10mA）。同时，该口也可设为相关功能引脚，如 ADC、PWM、外部中断。

P0 口的控制寄存器名为 P0CONH（E6H），P0CONL（E7H）。

可直接访问该口，进行读写。



注释：I/O 引脚有保护二极管通过 VDD 和 VSS

模 式	输入数据
输 出	D0
输 入	D1

图 9-2 P0 口电路图

P0 口控制寄存器

E6H，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7-.6]P0,P0.7/ADC7 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=A/D 转换输入（ADC7）；施密特触发器输入关闭

[.5-.4]P0,P0.6/ADC6/PWM 功能位

00=施密特触发器输入，带上拉电阻

01=相关功能（PWM 输出）

10=推挽式输出

11=A/D 转换输入（ADC7）；施密特触发器输入关闭

[.3-.2]P0, P0.5/ADC5 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=转换输入（ADC5）；施密特触发器输入关闭

[.3-.2]P0, P0.4/ADC4 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=转换输入（ADC4）；施密特触发器输入关闭

图 9-3　　P0 口控制寄存器（P0CONH）

P0 口控制寄存器

E7H，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7-.6]P0, P0.3/ADC3 功能位

00=施密特触发器输入

01=施密特触发器输入，带上拉电阻

10=推挽式输出

11=A/D 转换输入（ADC3）；施密特触发器输入关闭

[.5-.4]P0, P0.2/ADC2 功能位

00=施密特触发器输入

01=施密特触发器输入，带上拉电阻

10=推挽式输出

11=A/D 转换输入（ADC2）；施密特触发器输入关闭

[.3-.2]P0, P0.1/ADC1/INT1 功能位

00=施密特触发器输入/下降沿中断输入

01=施密特触发器输入；带上拉电阻下降沿中断输入

10=推挽式输出

11=A/D 转换输入（ADC1）；施密特触发器输入关闭

[.1-.0]P0, P0.0/ADC0/INT0 功能位

00=施密特触发器输入/下降沿中断输入

01=施密特触发器输入；带上拉电阻/下降沿中断输入

10=推挽式输出

11=转换输入（ADC0）；施密特触发器输入关闭

图 9-4　　P0 口控制寄存器（P0CONL）

P0 口控制寄存器

E8H，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7-.4] 没有用到

[.3]P0, P0.1/ADC1/INT1,中断使功能位
0=INT1 下降沿中断禁止
1=INT1 下降沿允许

[.2]P0, P0.1/ADC1/INT1,中断使能位
0=没有中断标志（当读此位时）
0=清除中断标志（当写此位时）
1=产生中断，置此位为 1（当读此位时）
1=没有作用（当写此位时）

[.1]P0, P0.0/ADC0/INT0,中断使能位
0=INT1 下降沿中断禁止
1=INT1 下降沿允许

[.0]P0, P0.0/ADC0/INT0，中断使能位
0=没有中断标志（当读此位时）
0=清除中断标志（当写此位时）
1=产生中断，置此位为 1（当读此位时）
1=没有作用（当写此位时）

图 9-5 P0 中断标志位（P0PND）

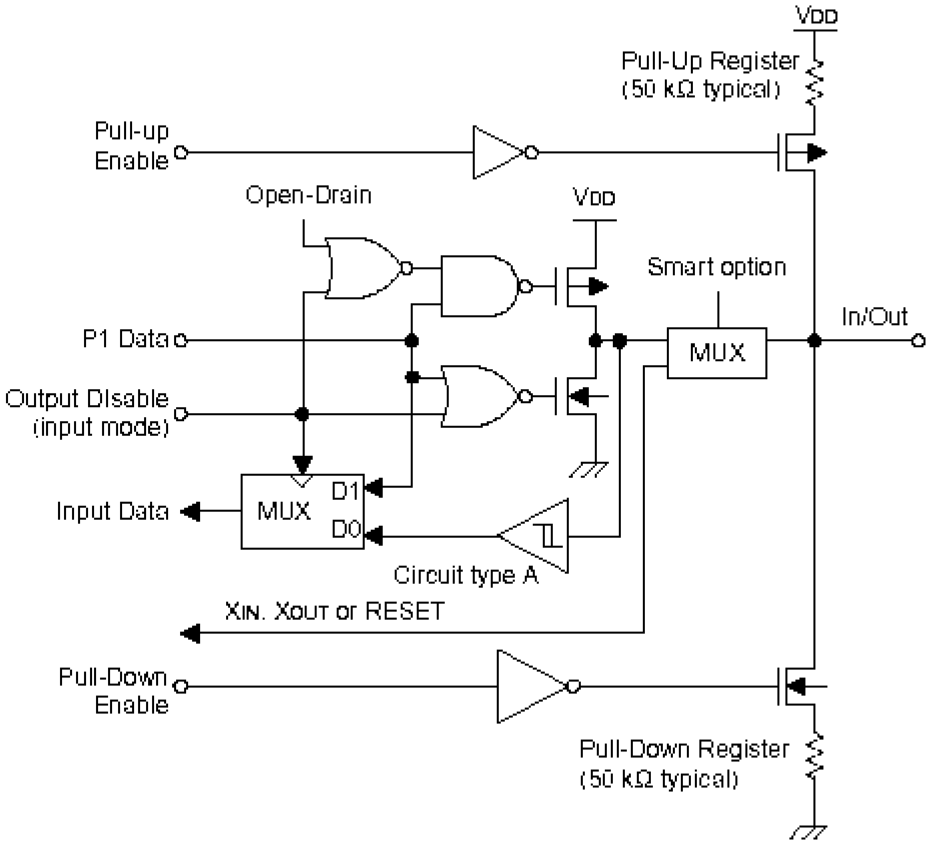
P1 口

P1 口是 3 位可设定的 I/O 口。你可以选择此口为通用 I/O 口（施密特触发器输入模式，推挽式输出模式，n 沟道开漏输出模式）。另外，你可以通过控制寄存器设定上拉电阻或下拉电阻。此口有较大的电流输出功能，能直接驱动 LED。

P1.0,P1.1 可以有 Smart Option 设定为晶振输入/输出，P1.2 可以设定为复位脚。

P1 口的控制寄存器名为 P1CON（E9H）。

可直接访问该口，进行读写。**当接外部晶振的时候，P1.0,P1.1 需设为输出模式以减少电流消耗。**



注释：I/O 引脚有保护二极管通过 VDD 和 VSS

模　式	输入数据
输　出	D0
输　入	D1

图 9-8　P2 口电路图

P1 口控制寄存器
E9H，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7]P1, P1.1 n 沟道开漏输出使能位
0=P1.1 为推挽式输出
1=P1.1 为 n 沟道开漏输出

[.6]P1, P1.0 n 沟道开漏输出使能位
0=P1.0 为推挽式输出
1=P1.1 为 n 沟道开漏输出

[.5-.4] 没有用到

[.3-.2]P1, P1.1 功能位
00=施密特触发器输入
01=施密特触发器输入，带上拉电阻
10=推挽式输出
11=施密特触发器输入，带下拉电阻

[.1-.0]P1, P1.1 功能位
00=施密特触发器输入
01=施密特触发器输入，带上拉电阻
10=推挽式输出
11=施密特触发器输入，带下拉电阻

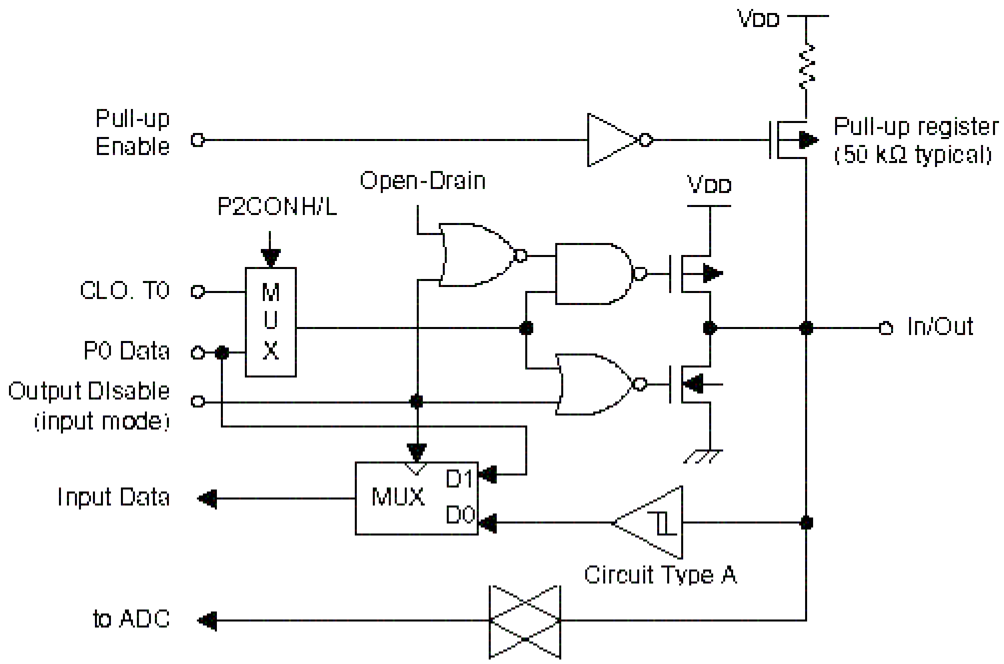
图 9-7　　P1 口控制寄存器

P2 口

P2 口是 7 位可设定的 I/O 口。你可以设定此口为通用 I/O 口（施密特触发器输入、推挽式输出、n 沟道开漏输出）。你也可以设定 P2 口的一些引脚为 ADC、CLO、T0 时钟输入。另外，也可以通过控制寄存器设定各引脚的上拉电阻。该口有较大的电流输出，可直接驱动 LED。

P2 口的控制寄存器名为 P2CONH（EAH），P2CONL（EBH）。

可直接访问该口，进行读写。



注释：I/O 引脚有保护二极管通过 VDD 和 VSS

模　式	输入数据
输　出	D0
输　入	D1

图 9-8　　P2 口电路原理图

P2 口控制寄存器

EAH，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7] 没有用到

[.6-.4]P2, P2.6/ADC8/CLO 功能位
000=施密特触发器输入，带上拉电阻
001=施密特触发器输入
01X=ADC 输入
100=推挽式输出
101=开漏输出，带上拉电阻
110=开漏输出
111=相关功能；CLO 输出

[.3-.2]P2, P2.5 功能位
00=施密特触发器输入，带上拉电阻
01=施密特触发器输入
10=推挽式输出
11=开漏输出

[.1-.0]P2, P2.4 功能位
00=施密特触发器输入，带上拉电阻
01=施密特触发器输入
10=推挽式输出
11=开漏输出

注释：　当噪声比较严重时，最好关闭 CLO

图 9-9　　P2 口控制寄存器

P2 口控制寄存器

EBH，R/W

.7	.6	.5	.4	.3	.2	.1	.0

[.7-.6]P2, P2.3 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=开漏输出

[.5-.4]P2, P2.2 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=开漏输出

[.3-.2]P2, P2.1 功能位

00=施密特触发器输入，带上拉电阻

01=施密特触发器输入

10=推挽式输出

11=T0 匹配输出

图 9-10 P2 口控制寄存器（P2CONL）

第十章　　　　定时器

概述

S3C9444/C9454 有 2 个定时器：8 位的 basic timer 和一个通用定时/计数器--T0

Basic Timer(BT)

Basic timer 有 2 个不同的功能：
--用作看门狗。当系统失灵时，复位系统。
--用于稳定晶振。当系统退出 STOP 状态或系统刚复位时，晶振不稳定，此定时器可延时稳定晶振。
BT 有以下几部分组成：
--时钟分频器（fosc/4096,fosc/1024,fosc/128）
--8 位 BT 计数器，BTCNT（DDH，只读）
--BT 控制寄存器，BTCON（DCH，读/写）

T0 定时器

T0 有以下几部分组成：
--时钟分频器（fosc/4096,fosc/1024,fosc/128）
--8 位 T0 计数器（T0CNT），8 位比较器，8 位数据寄存器（T0DATA）
--T0 控制寄存器（T0CON）

BASIC TIMER(BT)

BT 控制寄存器

BT 控制寄存器（BTCON）决定该定时器的输入时钟频率，清除计数值或分频器，禁止或使能看门狗功能。

复位时，BTCON 的值为 00H。“00H”将允许看门狗功能，选择定时器时钟频率为 fosc/4096。禁止看门狗功能，需要把‘1010B’写入 BTCON.7—BTCON.4。

在正常操作情况下，可以写’1’到 BTCON.1 清除 BT 计数器的值，也可以写’1’到 BTCON.0 清除 BT 定时器或 T0 定时器的分频器。

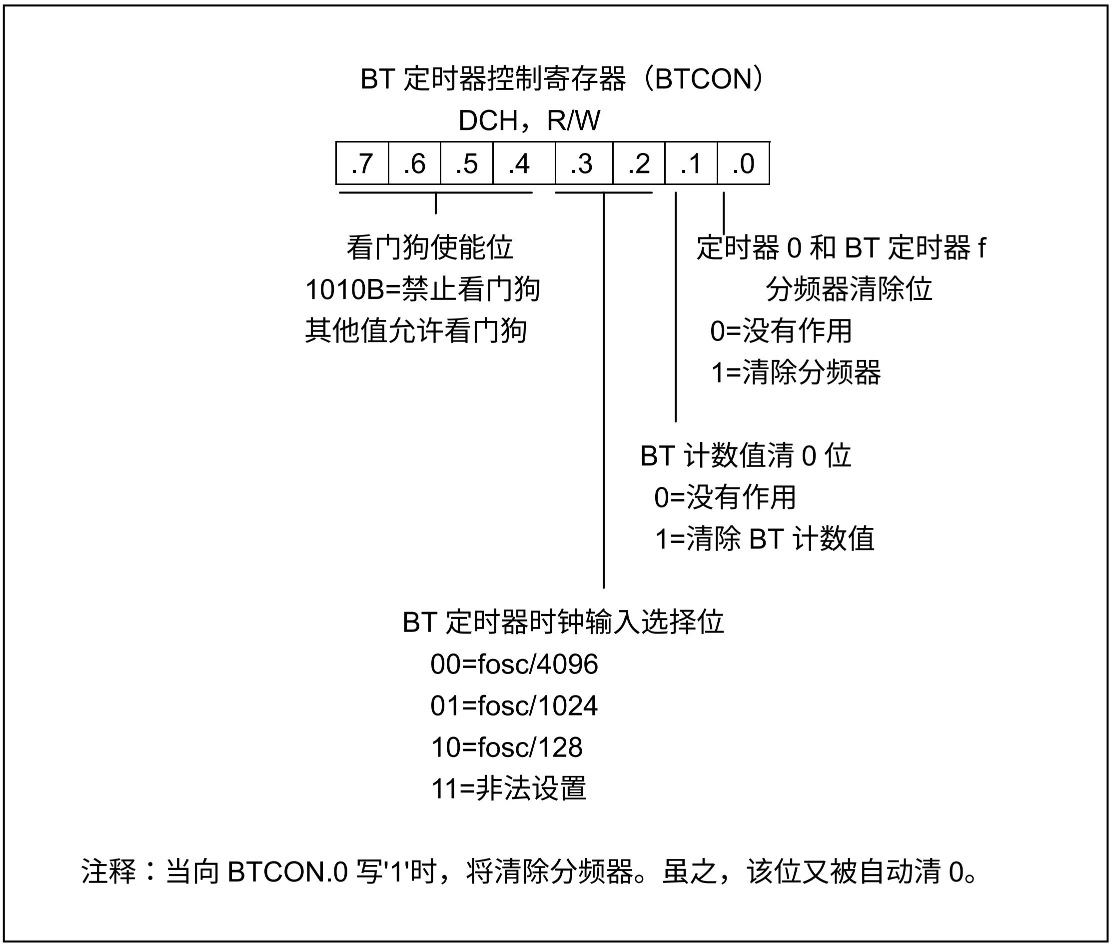


图 10-1　　BT 控制寄存器（BTCON）

BT 定时器功能描述

看门狗功能

写入 BTCON.7—BTCON.4 的值如果不是'1010B'，将会允许看门狗功能。当看门狗计数溢出之后，就会复位 MCU。复位将清 BTCON 为'00H'。同时，允许看门狗。在正常情况下，应该禁止看门狗溢出，所以每隔一段时间要清除看门狗计数值。

如果由于噪声或其他原因造成系统失灵，则 BT 计数溢出后，将会复位系统。

晶振稳定功能

当复位操作或系统退出 STOP 状态时，BT 定时器可以用来稳定晶振。当系统从 STOP 状态退出时，晶振起振，BTCNT 以 fosc/4096 或预先设定的值计数，当 BTCNT.4 被置起的时候，将产生一个信号，说明晶振已经稳定，CPU 开始在稳定的晶振下开始工作。

当系统从 STOP 状态退出时，以下的情况就会发生：

1　复位或外部中断产生，使系统退出 STOP 状态。
2　如果是复位操作，则 BT 以 fosc/4096 的时钟频率计数，如果是外部中断，则以预先设定的值计数
3　BTCNT.4 被置起，晶振稳定。
4　CPU 开始正常操作。

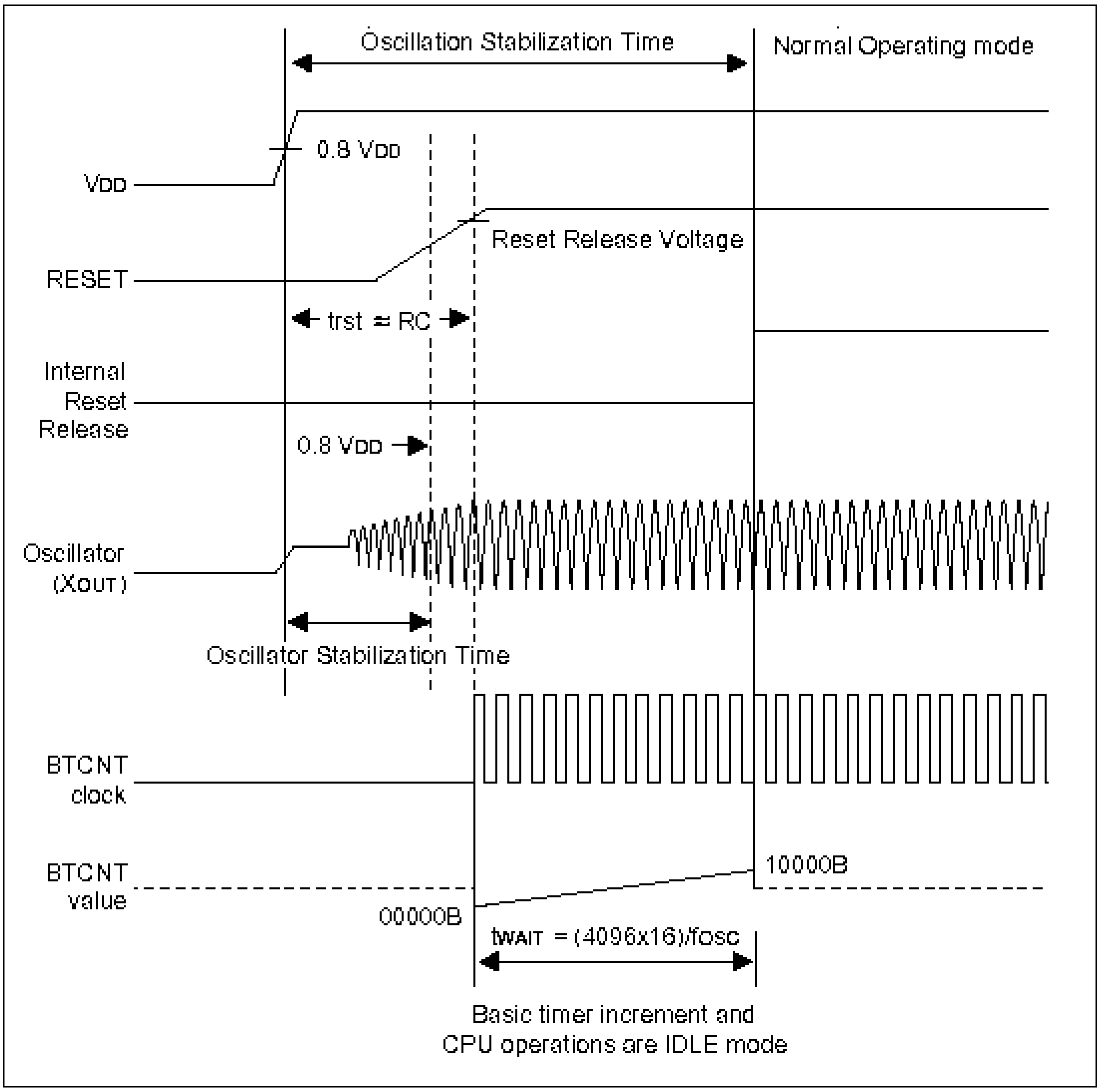
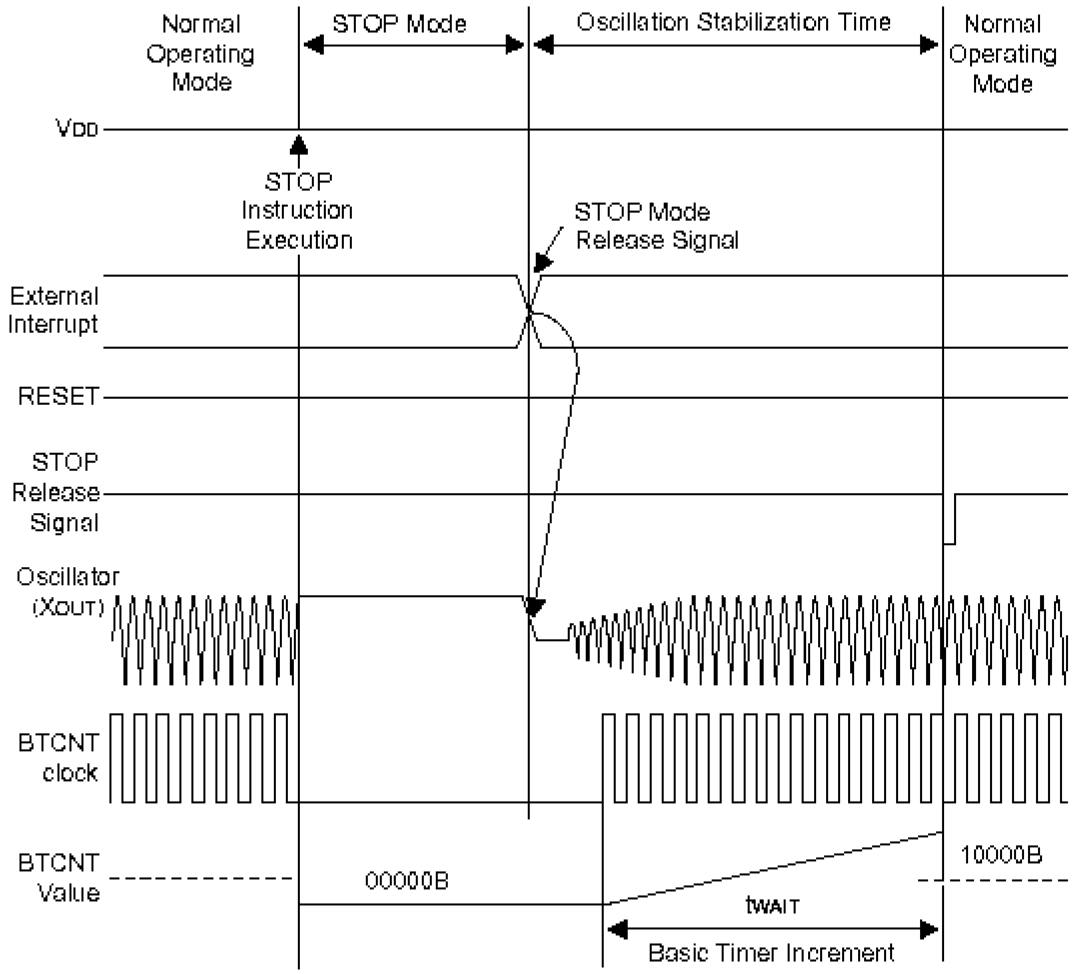


图 10-2　　复位后晶振稳定时间

注释：等待晶振稳定的时间 Twait=(4096×16)/fosc

Trst=RC



注释：Twait 的时间由具体的 BTCON 设置决定。

BTCIN.3	BTCON.2	Twait	Twait(fosc=10M)
0	0	(4096×16)/fosc	6.55ms
0	1	(1024×16)/fosc	1.64ms
1	0	(128×16)/fosc	0.2ms
1	1	非法设置	--

图 10-3　　从 STOP 状态退出晶振的稳定时间

☞BT 编程实例

```
                    ORG       0000H
                    VECTOR    00H,INT_9454
;------------------------<<Smart Option>>
                    ORG       003CH
                    DB        00H             ;必须清为 0
                    DB        00H             ;必须清为 0
                    DB        0E7H            ;允许低电压复位
                    DB        03H             ;内部 RC 震荡 3.2M(VDD=5V)
;-------------------------<<初始化>>
                    ORG       0100H
        RESET:      DI
                    LD        CLKCON,#00011000B
                    LD        SP,#0C0H
                    .
                    .
                    LD        BTCON,#02H
                    .
                    .
                    .
                    EI
;---------------------------<<主程序>>
        MAIN:       NOP
                    LD        BTCON,#02H
                    .
                    .
                    JR        T,MAIN
;---------------------------<<中断服务程序>>
        INT_9454:   .
                    .
                    .
                    IRET
                    .
                    .
                    .END
```

T0 定时器

T0 控制寄存器（T0CON）

T0 控制寄存器可以选择 T0 工作模式、输入时钟频率、使能 T0 匹配中断以及清除 T0 计数器的值。同时，也有 T0 匹配中断标志位。

复位操作后，T0CON 为’00H’。这是 T0 处于周期定时中断模式，时钟频率为 fosc/4096，禁止 T0 匹配中断。在任何时候，写’1’到 T0CON.3 都可以清除 T0 的计数值。

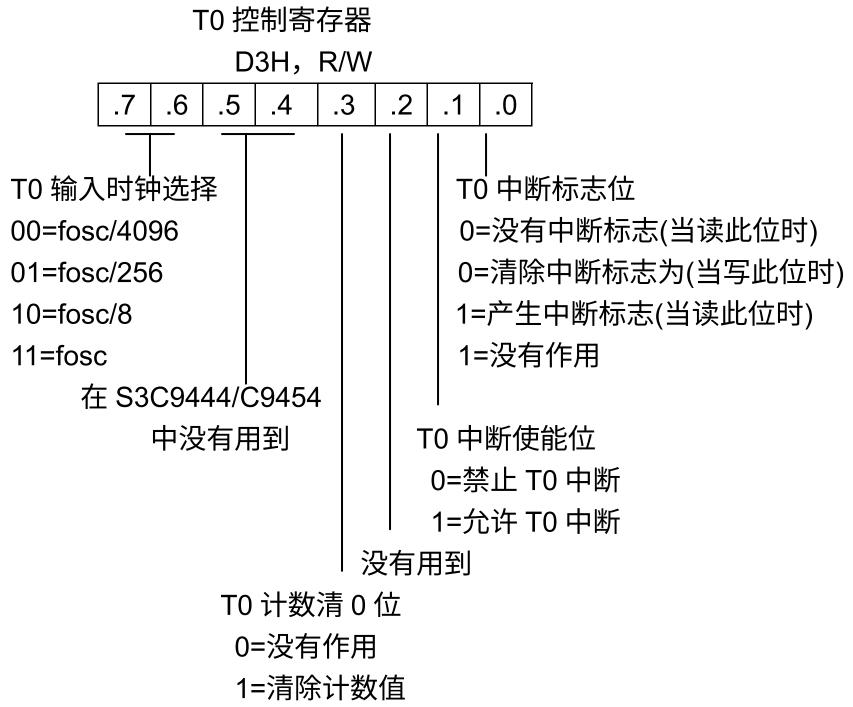


图 10-4　　T0 控制寄存器（T0CON）

T0 定时器功能描述

周期定时模式

在周期定时模式下，当 T0 计数器的值与事先写道 T0DATA 中的数据相等时产生匹配中断信号，同时清除 T0 定时器的计数值。例如：假如写到 T0DATA 中的值为'10H',T0 计数器会一直计数值到'10H'。这时，产生中断，计数值清 0，继续计数。

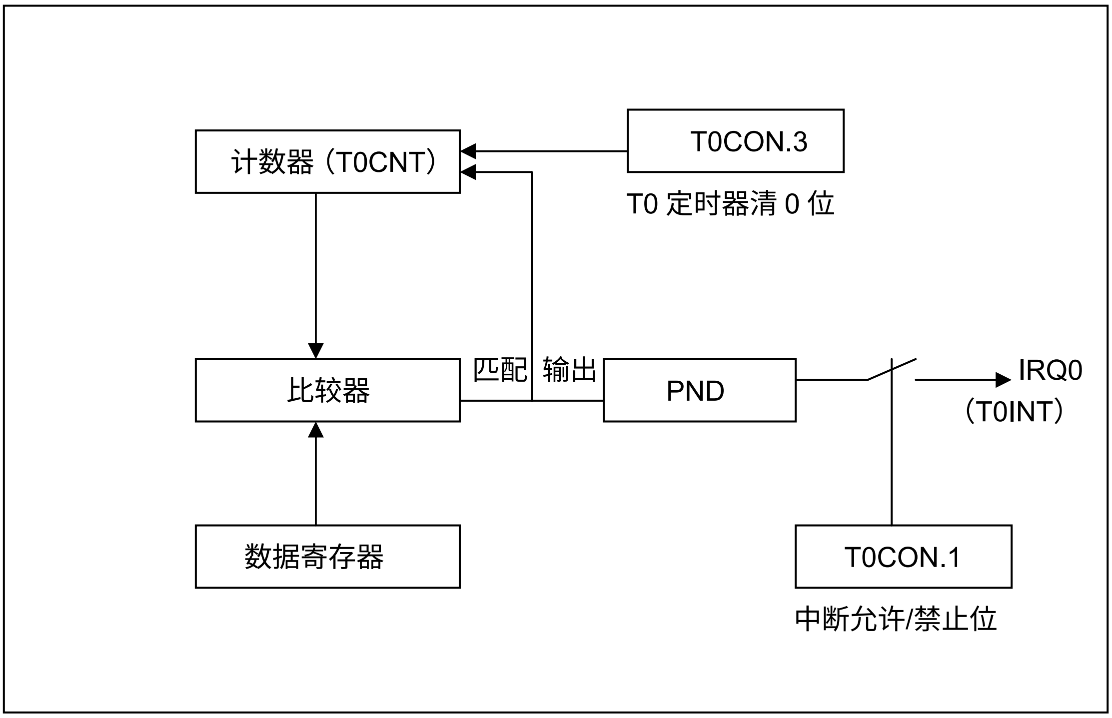


图 10-5　　T0 定时器功能简化图

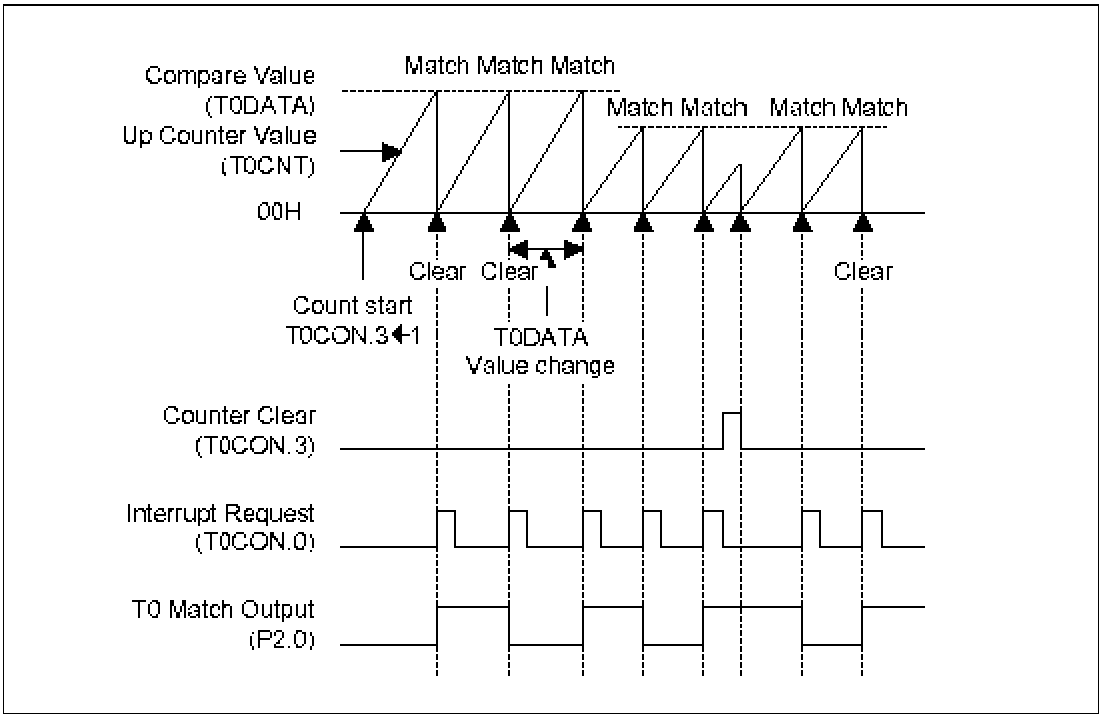


图 10-6 T0 定时器定时原理图

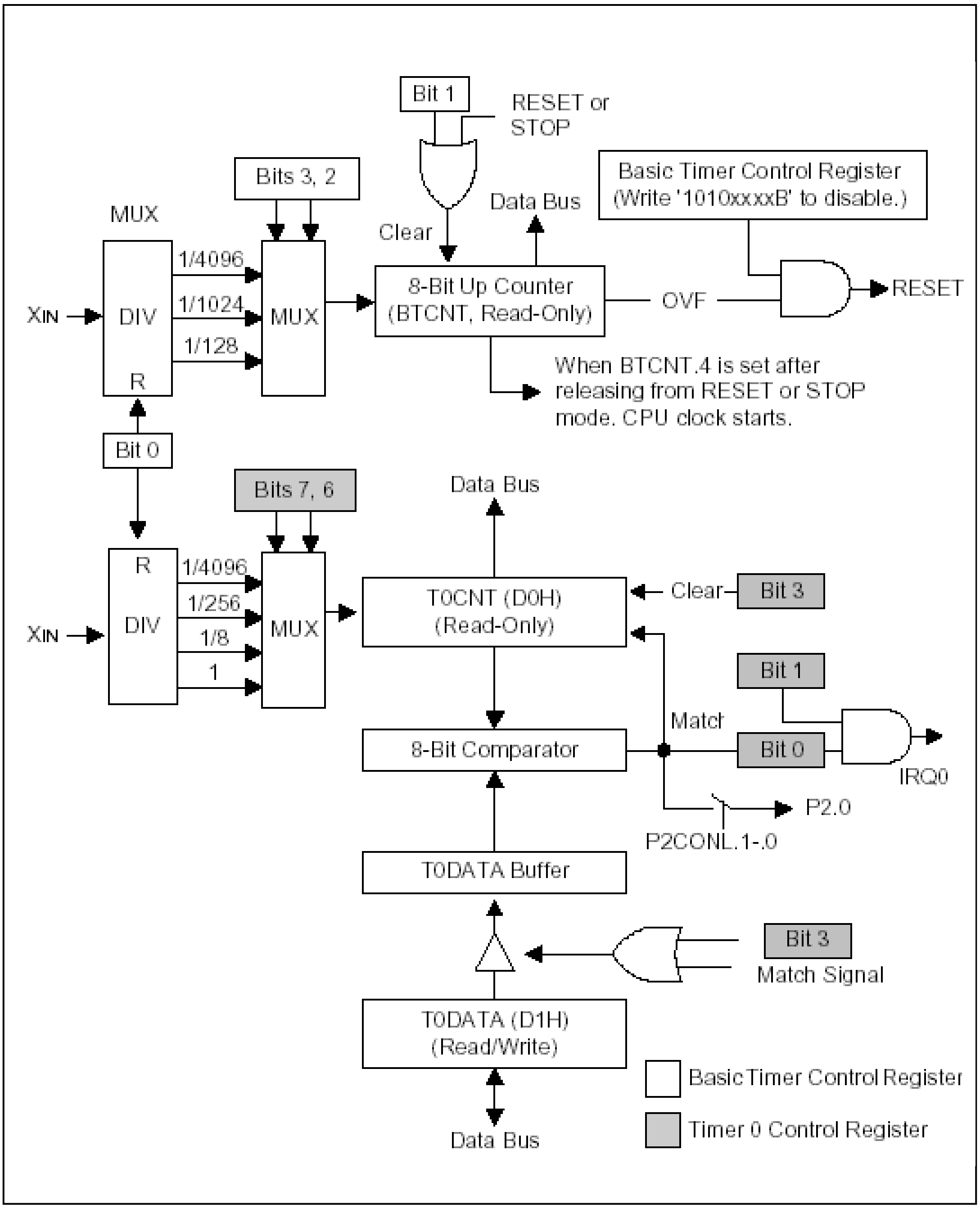


图 10-7　　BT 定时器和 T0 定时器的原理图

☞ 编程实例

```
                ORG       0000H
                VECTOR    00H,INT_9454          ;中断向量地址

                ORG       003CH
                DB        00H                   ;必须初始化为 0
                DB        00H                   ;必须初始化为 0
                DB        0E7H                  ;允许低电压复位
                DB        03H                   ;内部震荡 RC3.2M

                ORG       0100H
RESET:          DI                              ;禁止中断
                LD        BTCON,#10100011B      ;禁止看门狗
                LD        CLKCON,#00011000B     ;选择 CPU 时钟
                LD        SP,#0C0H
                LD        P0CONH,#10101010B     ;P0 口推拉输出
                LD        P0CONL,#10101010B     ;
                LD        P1CON,#00001010B      ;P1.1-P1.0 为推拉式输出
                LD        P2CONH,#01001010B     ;P2 为推拉式输出
                LD        P2CONL,#10101010B

;-------------------------<<T0 设置>>
                LD        T0DATA,#50H
                LD        T0CON,#01001010B
                .
                .
                EI

;------------------------<<主函数>>
          MAIN: NOP
                .
                .
                CALL      LED_DISPLAY
                .
                .
```

```
                CALL        JOB
                .
                .
                JR          T,MAIN
LED_DISPLAY:    NOP
                .
                .
                RET

JOB:            NOP
                .
                .
                RET

;--------------------<<中断服务程序>>
INT_9454:       TM          T0CON,#00000010B
                JR          Z,NEXT_CHK1

                TM          T0CON,#00000001B
                JP          NZ,INT_TIMER0

NEXT_CHK1:      .
                .
                IRET

INT_TIMER0:     .
                .
                .
                AND         T0CON,#11110110B
                IRET
                .END
```

第十一章　　　脉宽调制

概述

S3C9444/C9454 内部有 8 位 PWM 电路。该电路有 PWM 控制寄存器（PWMCON）控制。PWM 是一个自增 8 位计数器。设 PWMCON.2 为'1'，则允许 PWM 并开始计数。如果停止 PWM，则该计数器保留当前计数值。当重新开始计数后，又从原先计数值开始计数。设 PWMCON.3 为'1'，则清除计数值。

PWM 的时钟频率输入也可以选择 fosc/64,fosc/8,fosc/2,fosc/1.通过设置 PWMCON.6-.7，可以选择 PWM 的时钟频率。

PWM 功能描述

PWM

8 位 PWM 有以下几个部分：

— 6 位比较器和周期扩展电路

— 6 位数据比较寄存器（PWMDATA.7-.2）

— 2 位数据扩展位（PWMDATA.1-.0）

— PWM 输出脚（P0.6/PWM）

PWM 计数器

PWM 的输出频率，是由计数器的高 6 位与 PWM 的数据寄存器（PWMDATA.7-.2）决定的。为了完成更高端的性能，计数器的低 2 位用来扩展周期。计数器的末 2 位，用于决定 PWM 输出特定周期占空比的'延伸'。计数器的末 2 位，将会与 PWMDATA.1-.0 比较。

PWM 数据寄存器

PWM 数据寄存器的地址为 F2H，它决定 PWM 的输出频率。为得到合适的脉宽输出，可以设定 6 位数据比较器和 2 位周期'延伸'控制位。设置 PWMCON.2 为'1'，将开始计数或者在原基础上继续计数。

复位操作将禁止 PWM 输出。当停止 PWM 时，PWM 计数器的值保留在计数器中。

PWM 时钟频率

PWM 的输出频率特性是以 fosc 的频率为基础的。PWM 的时钟是由 PWM.6-.7 决定的。

表 11-1　　PWM 控制寄存器和数据寄存器

寄存器名	符　　号	地　　址	功　　　能
PWM 数据寄存器	PWMDATA.7-.2	F2H.7-.2	6 位 PWM 频率值
	PWMDATA.1-.0	F2H.1-.0	2 位周期‘延伸’值
PWM 控制寄存器	PWMCON	F3H	PWM 计数启动/停止，PWM 时钟选择

PWM 功能描述

当 PWM 的 6 位计数器的值等于 PWM 数据寄存器高 6 位（PWMDATA.7-.2）的值时，PWM 输出低电平。如果 PWMDATA.7-.2 不为 0，则 6 位计数溢出后，PWM 输出高电平。就是这样来决定 PWM 输出的占空比。

计数器的高 2 位（.7-.6）与 PWMDATA.1-.0 比较决定‘延伸’周期。具体的‘延伸’周期如下表。

例如，如果扩展位数据寄存器的值为’01B’，则第 2 个机器周期会比其他的 3 个机器周期多一个脉宽。如果占空比为 50％，则拉伸后占空比约为 51％。如果写’10B’到数据寄存器的扩展位，所有奇数脉宽都会加宽一个脉宽。如果写’11B’到数据寄存器扩展位，则所有机器周期都会增加一个脉宽，除了第 4 个机器周期。PWM 输出先到一个输出缓存，然后再输出。

表 11-2　　PWM‘延伸’与对应值

PWMDATA 位（BIT1-BIT0）	‘延伸’周期
00	--
01	2
10	1，3
11	1，2，3

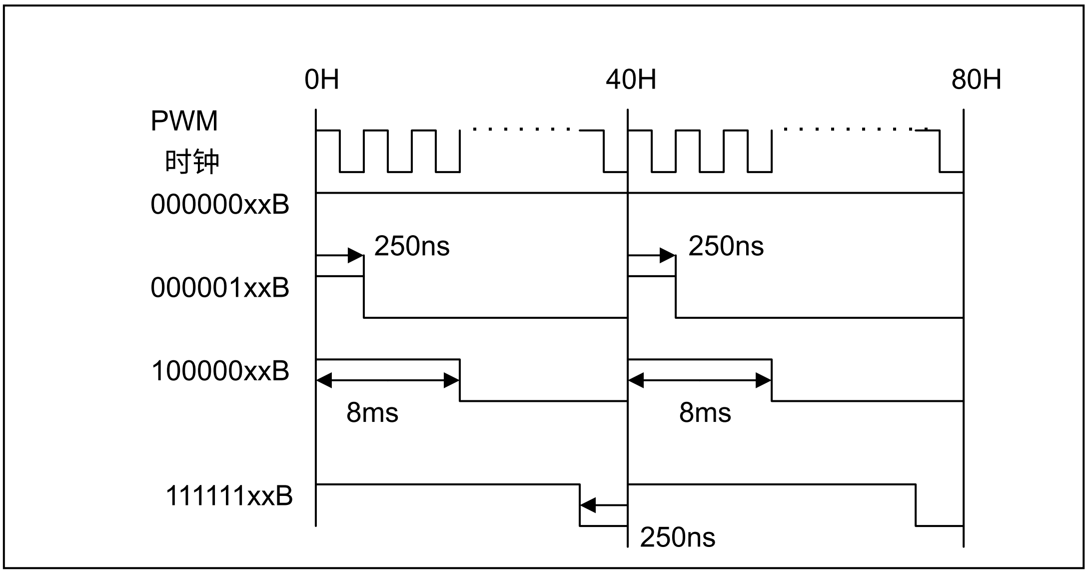


图 11-1 8 位 PWM 输出波形

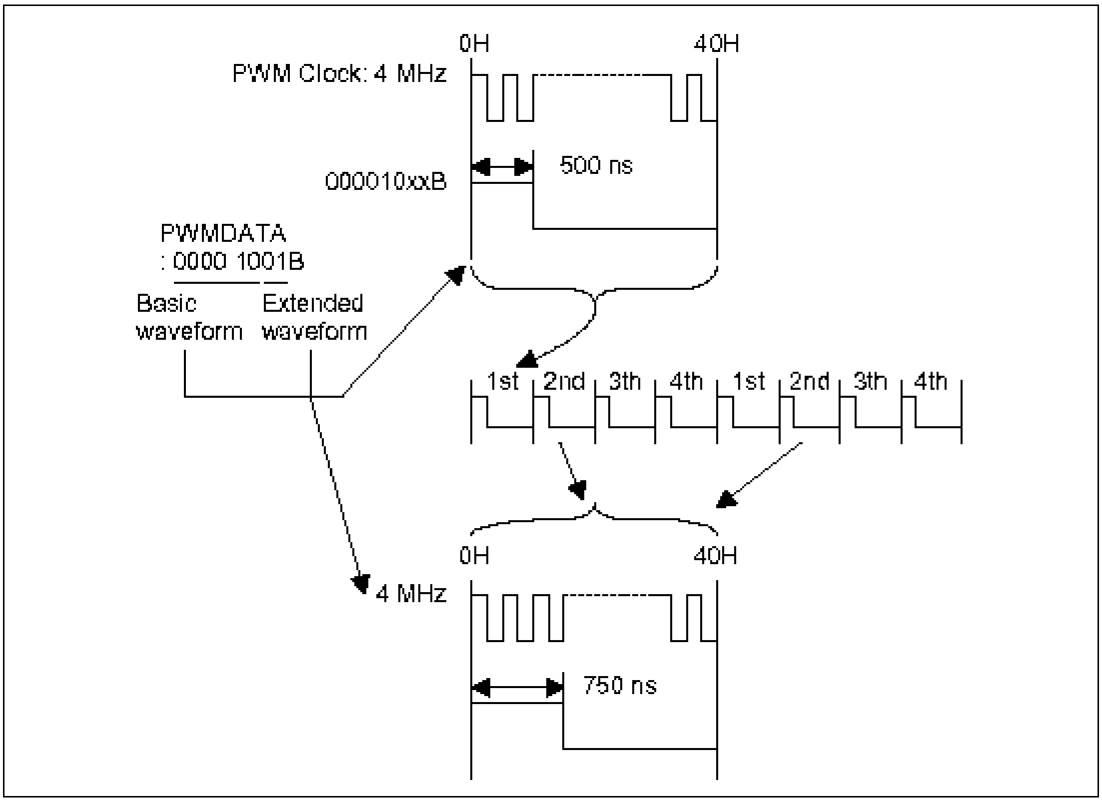


图 11-2 8 位'延伸'波形

PWM 控制寄存器

PWM 控制寄存器的地址为 F3H。通过设置 PWMCON 寄存器可以控制以下功能：

— PWM 时钟选择
— PWM 数据重装周期
— PWM 计数清 0
— PWM 计数停止/开始操作
— PWM 溢出中断控制

复位操作清除 PWMCON 为 00H，禁止 PWM 输出。

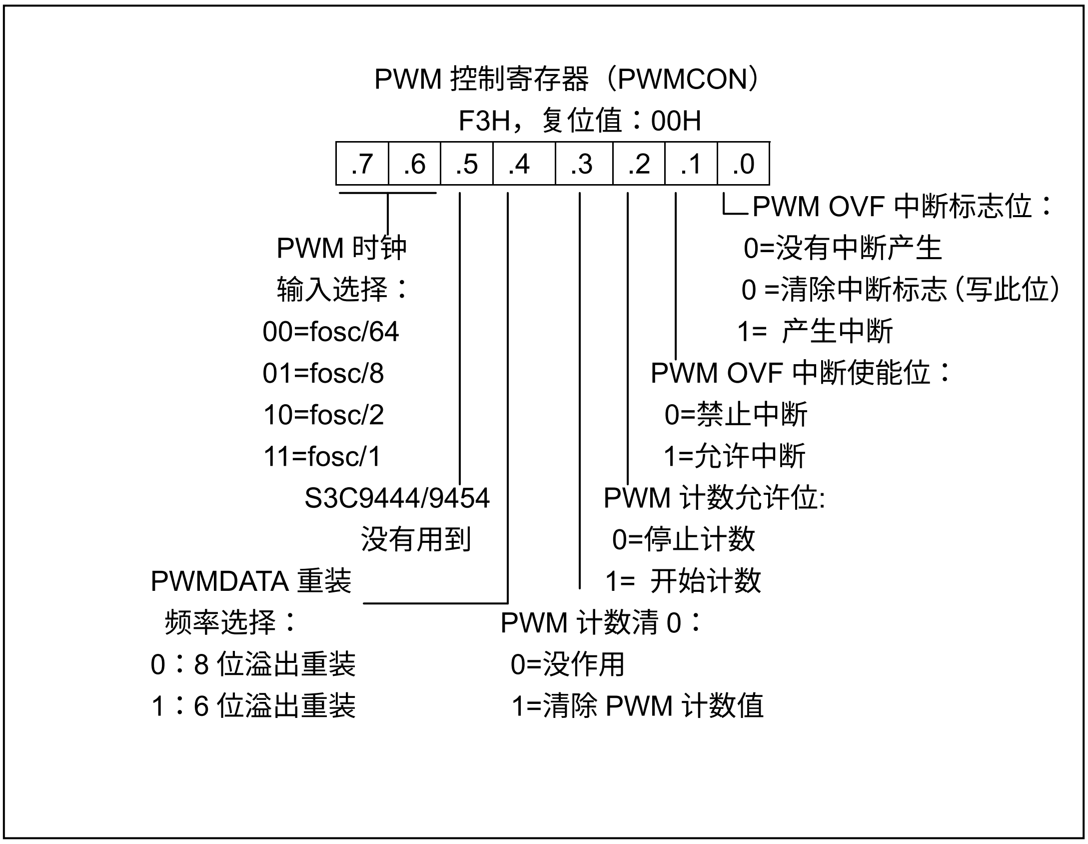


图 11-3　　PWM 控制寄存器（PWMCON）

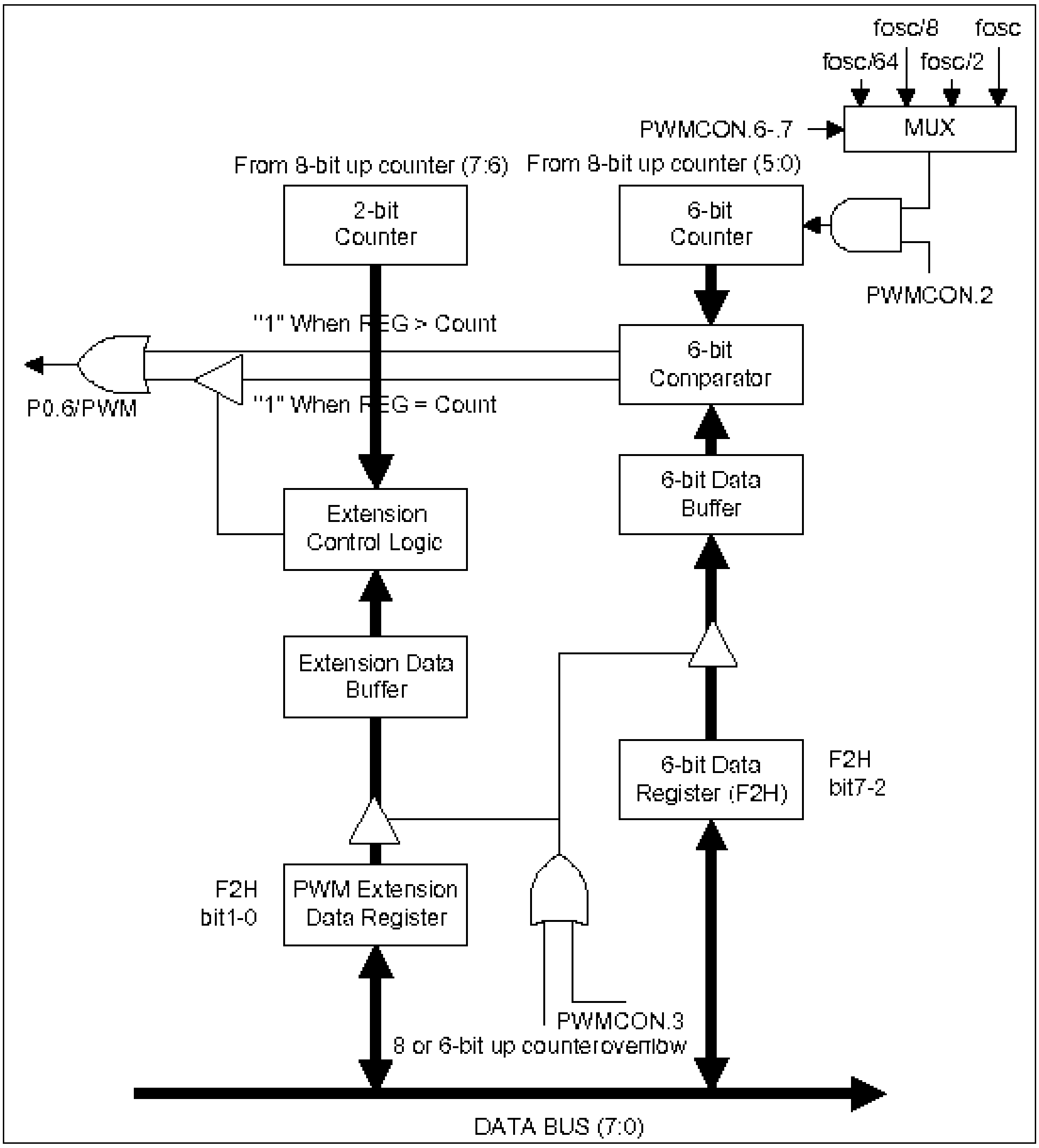


图 11-4 PWM 功能原理图

编程实例：

```
;------------------------<<中断向量地址>>
                        ORG         0000H
                        VECTOR      00H,INT_9454
;------------------------<<Smart Option>>
                        ORG         0100H
                        DB          00H
                        DB          00H
                        DB          0E7H
                        DB          03H
;-------------------------<<初始化系统>>
                        ORG         0100H
        RESET:          DI
                        LD          BTCON,#10100011B
                        LD          PWMCON,#00000110B
                        .
                        LD          P0CONH,#10011010B
                        LD          PWMCON,#00000110B
                        .
                        EI
;-------------------------<<主函数>>
        MAIN:           .
                        JR          T,MAIN
;------------------------<<中断服务>>
       INT_9454:        TM          PWMCON,#00000010B
                        JR          Z,NEXT_CHK1
                        TM          PWMCON,#00000001B
                        JP          NZ,INT_PWM
       NEXT_CHK1:       .
                        IRET
       INT_PWM:         .
                        .
                        AND         PWMCON,#11110110B
                        IRET
                        .END
```

第十二章　　A/D 转换

概述

S3C9444/C9454 内部有 9 路 A/D,可以把连续变化的模拟信号转换为 10 位数字量。模拟量的值在 VDD 和 VSS 之间。A/D 转换有以下几个部分：

— 模拟比较器
— D/A 转换逻辑
— ADC 控制寄存器（ADCON）
— 9 路模拟信号输入端（ADC0-ADC8）
— 10 位 A/D 转换结果寄存器（ADDATAH/L）

为了开始一个模数转换，需要在 A/D 控制寄存器（ADCON）中选择一路模拟输入量，同时启动 A/D 转换。A/D 控制寄存器的地址为 F7H。

在一个正常转换过程中，ADC 逻辑电路设置连续近似转换结果寄存器的值为 200H（10 位转换结果的一半）。这个寄存器在每次转换时会自动刷新。通过操作 ADCON 寄存器的 ADCON.7-.4 可以动态选择多路 AD 转换。设置 ADCON.0 位'1'，则启动 A/D 转换。当 A/D 转换结束时，控制寄存器 ADCON.3 自动置'1'，即 EOC 被置'1'。AD 转化结果装入 ADDATA 寄存器。A/D 转换电路进入空闲状态。在开始下一次转换之前，上一次 A/D 转换的数据应被读出。

A/D 转换输入端在不用 AD 转化的时候，可以用作数字量输入口。

注释：ADC 电路内部没有样本保存电路，所以在每次 AD 转换时，此路模拟信号输入端的波动要非常小，否则，转换结果可能会不正确。

A/D 转换控制寄存器

A/D 控制寄存器的地址为 F7H，给寄存器有 4 个功能：

— ADCON.7-.4 选择模拟量输入端（ADC0-ADC8）
— ADCON.3 标志 AD 转换状态
— ADCON.2-.1 选择转换速度
— ADCON.0 启动 AD 转换

一次只能转换一路模拟量。A/D 控制寄存器功能如下：

A/D 转换控制寄存器

F7H，R/W

.7	.6	.5	.4	.3	.2	.1	.0

A/D 转换模拟信号输入选择位（.7–.4）：
0000=ADC0(P0.0)
0001=ADC1(P0.1)
0010=ADC2(P0.2)
0011=ADC3(P0.3)
0100=ADC4(P0.4)
0101=ADC5(P0.5)
0110=ADC6(P0.6)
0111=ADC7(P0.7)
1000=ADC8(P2.6)
1001=内部接地
…..
1111=内部接地

A/D 转换结束标志位（.3）：
0=A/D 转换正在进行
1=A/D 转换结束

A/D 转换速度选择位（.2–.1）：
00=fosc/16(fosc<10MHZ)
01=fosc/8(fosc<10MHZ)
10=fosc/4(fosc<10MHZ)
11 =fosc/1(fosc<4MHZ)

A/D 转换开始（.0）：
0=没有作用
1=A/D 转换开始

注释：ADC 转换的最大时钟为 4MHZ

图 12-4　　A/D 转换控制寄存器

内部比较电压

在 A/D 转换中，输入模拟信号要与内部参考电压进行比较。输入模拟信号的电压范围应在 VDD 至 VSS 之间。不同的比较电压是由内部电阻网络分压产生的，开始的比较电压一般为 1/2VDD。

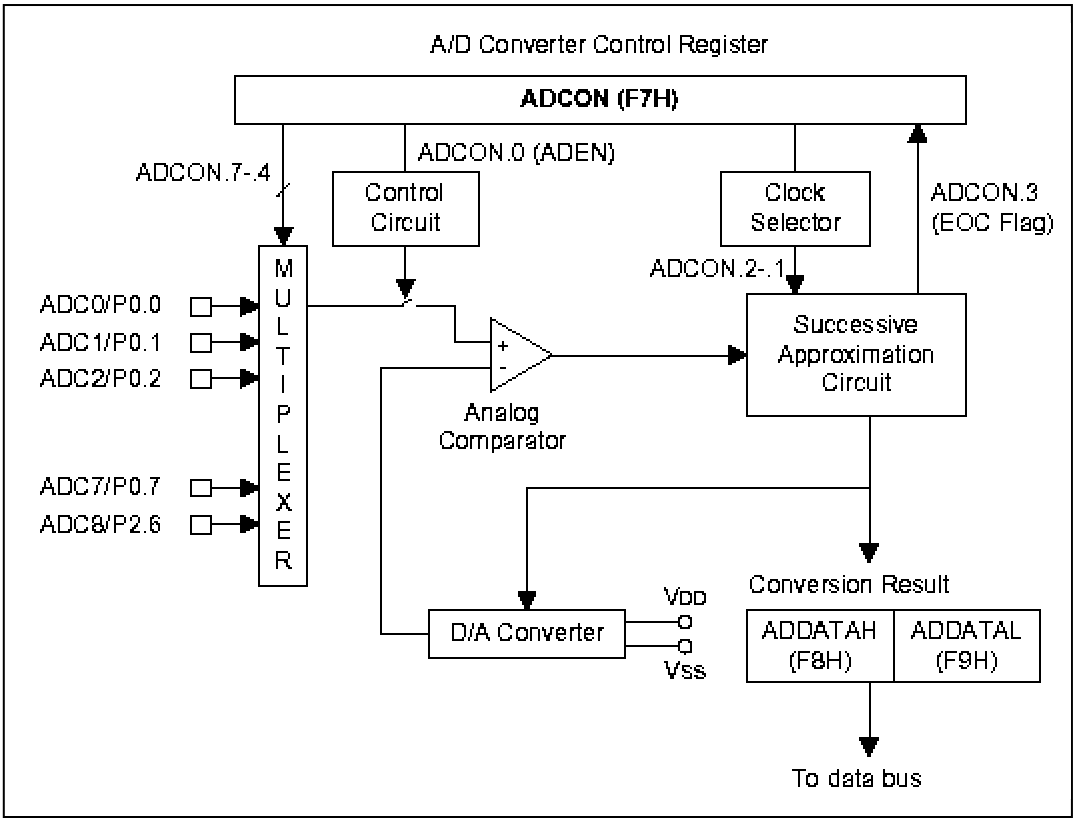


图 12-2 A/D 转换电路原理图

ADDATAH

MSB LSB

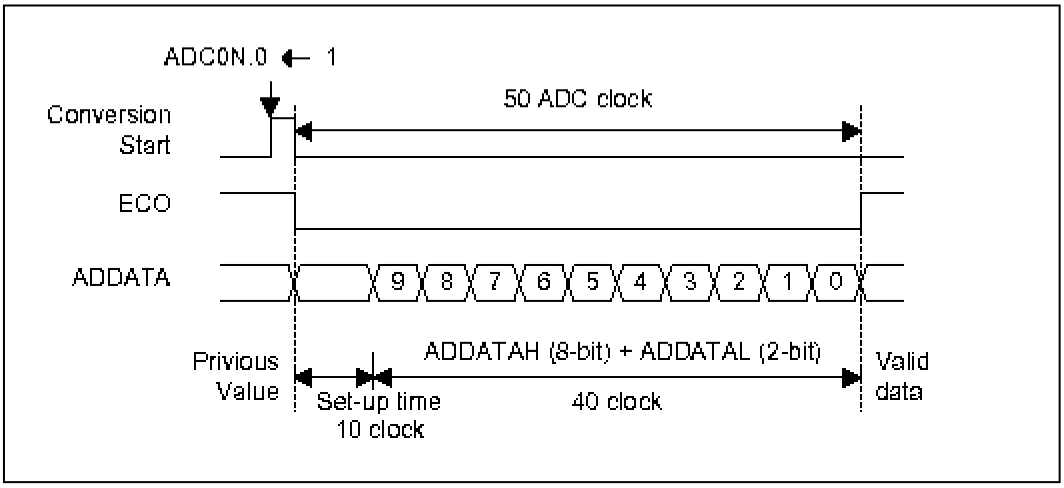
.7	.6	.5	.4	.3	.2	.1	.0

ADDATAL

MSB LSB

.7	.6	.5	.4	.3	.2	.1	.0

图 12-3 A/D 转换数据寄存器（ADDATAH/L）



A/D 转换时间

A/D 转换一位需要 4 个时钟，建立 A/D 转换需要 10 个时钟，因此 A/D 转换完 10 位需要 50 个时钟。在 10M 频率下，一个时钟周期为 400ns（4/fosc)。A/D 转换所需要的时间为：

4 时钟/位×10 位+建立时间（10 个时钟）=50 个时钟
50 个时钟×400ns=20μs （fosc=10M）
1 个时钟周期=4/fosc

内部 A/D 转换过程

1 模拟信号输入应该在 VSS 至 VDD。
2 设置 P0CONH,P0CONL 和 P2CONH 寄存器的值，可以设置模拟输入口的输入模式。
3 在开始转换之前，必须要选定一路模拟信号（通过设置 ADCON 控制寄存器)。
4 A/D 转换结束之后，将置起 EOC 标志位，至此位为‘1’，通过检测此位，可以确定 A/D 转换是否进行完毕。
5 A/D 转换完毕之后，相应的数据存放在 ADDATAH（8 位）和 ADDATAL（2 位）寄存器中。转换完毕之后，ADC 转换模块进入空闲状态。
6 数字结果可以从 ADDATAH 和 ADDATAL 中读出。

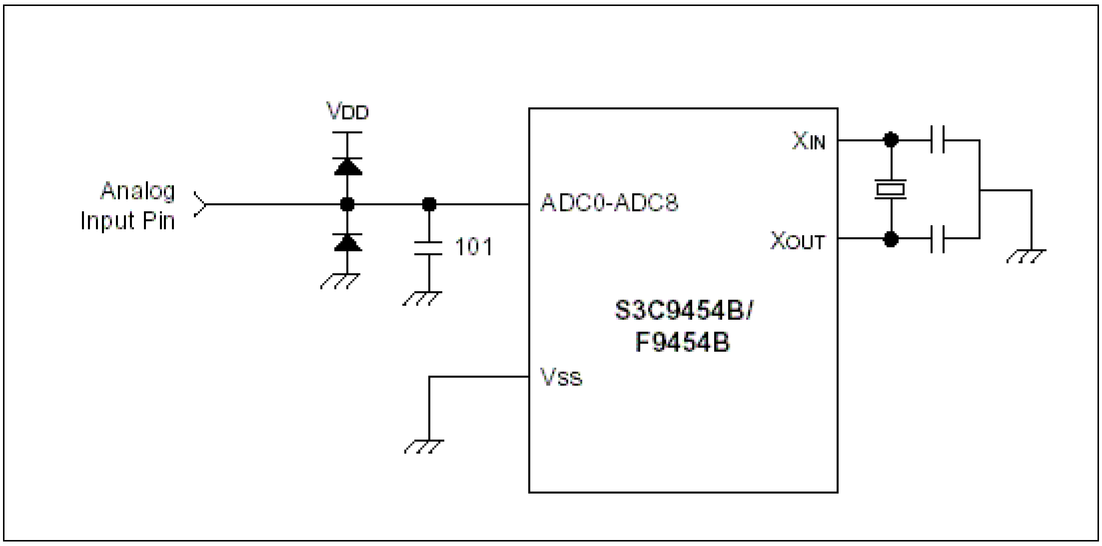


图 12-5 典型 A/D 转换电路

第13章　　　电气参数

概述

在这一章，我们将以图表或表格的方式向您提供 S3C9444/C9454 的电气参数：

— 芯片物理特性绝对变化范围
— 直流电气特性
— 交流电气特性
— 振荡器特性
— 振荡器稳定时间
— 操作电压范围
— 施密特触发器输入特性
— 在冻结模式（STOP）下数据保存所需电压
— A/D 转换电气特性
— LVR 电路特性
— LVR 复位时间

表 13-1　　芯片物理特性绝对变化范围

（T_A=25℃）

参　　数	标　号	条　　件	范　　围	单　位
供 电 电 压	VDD	-	-0.3---+6.5	V
输 入 电 压	V_I	所 有 I/O 口	-0.3---VDD+0.3	V
输 出 电 压	V_O	所 有 输 出 口	-0.3---VDD+0.3	mA
I/O 口输出电流	I_{OH}	单个 I/O 口工作时	-25	
		所有 I/O 口工作时	-80	
I/O 口灌电流	I_{OL}	单个 I/O 口工作时	+30	mA
		所有 I/O 口工作时	+150	
工 作 温 度	T_A	-	-40---+85	℃
储 藏 温 度	T_{STG}	-	-65---+150	℃

表 13-2 直流电气特性（T_A=-40°C到+85°C，VDD=2.0V~5.5V）

参 数	标号	条	件	最小值	典型值	最大值	单位
输 入高电压	V_{IH1}	P0,1,2,RESET	V_{DD}=2.0~5.5V	0.8V_{DD}	--	V_{DD}	V
	V_{IH2}	X_{IN}和X_{OUT}		V_{DD}-0.1			
输 入低电压	V_{IL1}	P0,1,2,RESET	V_{DD}=2.0~5.5V	--	--	0.2V_{DD}	V
	V_{IL2}	X_{IN}和X_{OUT}				0.1	
输 出高电压	V_{OH}	I_{OH}=-10mA P0,1,2	V_{DD}=4.5~5.5V	V_{DD}-1.5	V_{DD}-0.4	--	V
输 出低电压	V_{OL}	I_{OL}=25mA P0,1,2	V_{DD}=4.5~5.5V	--	0.4	2.0	V
输入漏电 流(1)	I_{LIH1}	除了 I_{LIH2} 所有输入	V_{IN}=V_{DD}	--	--	1	μA
	I_{LIH2}	X_{IN}和X_{OUT}	V_{IN}=V_{DD}			20	
输 入漏电流(1)	I_{LIL1}	除I_{LIH2},RESET所有输入	V_{IN}=0V	--	--	-1	μA
	I_{LIL2}	X_{IN}和X_{OUT}	V_{IN}=0V			-20	
输出漏电流(2)	I_{LOH}	所有的输出引脚	V_{OUT}=V_{DD}	--	--	2	μA
输 出漏电流(1)	I_{LOL}	所有的输出引脚	V_{OUT}=0V	--	--	-2	μA
$R^{(1)}$	R_P	V_{IN}=0V,P0,1,2	V_{DD}=5V	25	50	100	KΩ
$R^{(2)}$	R_P	V_{IN}=0V,P1	V_{DD}=5V	25	50	100	
供电电流	I_{DD1}	在 10M 频率下正常工作模式	V_{DD}=4.5~5.5V	--	5	10	mA
		在 3M 频率下	V_{DD}=2.0V		2	5	
	I_{DD2}	在 10M 频率下空闲工作模式	V_{DD}=4.5~5.5V	--	2	4	
		在 3M 频率下	V_{DD}=2.0V		0.5	1.5	
	I_{DD3}	冻 结 状 态（STOP）	V_{DD}=4.5~5.5V（禁止 LVR）	--	0.1	5	μA
			V_{DD}=4.5~5.5V (允许 LVR)		100	200	
			V_{DD}=2.6V（允许 LVR）		30	60	

注释：1　在冻结状态（STOP）和空闲状态（IDLE）的供电电流中，不包括 A/D 转换。
2　(1) 表示低电平
3　(2) 表示高电平
4　R(1) 表示上拉电阻
5　R(2) 表示下拉电阻

表 13-3　　交流电气特性

（T_A=-40°C到+85°C，VDD=2.0V~5.5V）

参　数	标 号	测试条件	最　小值	典 型 值	最 大 值	单　位
中断输入最低宽度	t_{INTL}	INT0，INT1 V_{DD}=5V±10％	--	200	--	nS
复位输入最低宽度	t_{RST}	输入 V_{DD}=5V±10％	--	1	--	μS

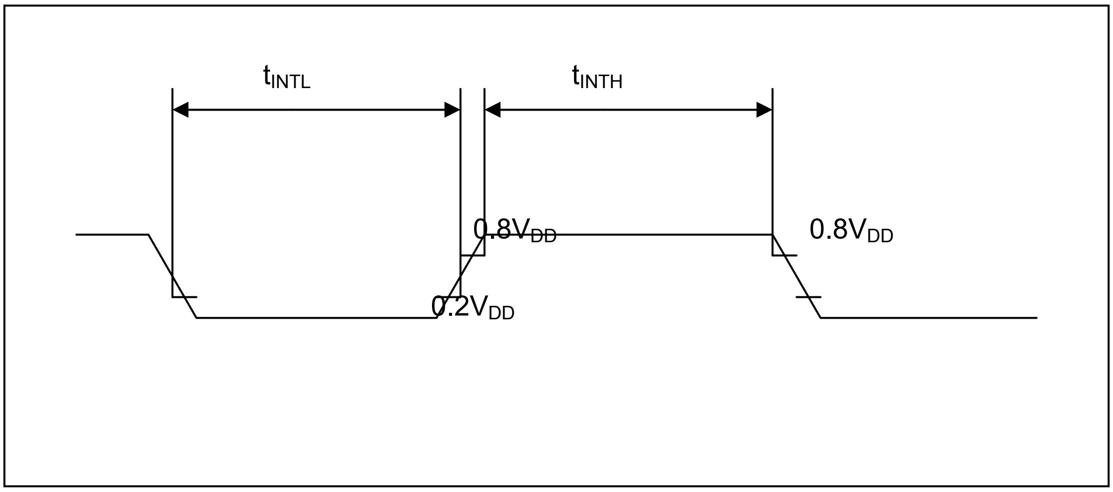


表 13-4　　振荡器特性

（T_A=-40°C到+85°C）

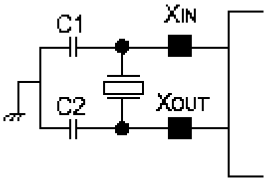
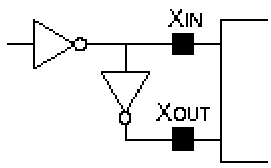
振　荡器	时钟电路	测　试　条　件	最　小值	典型值	最　大值	单位
石　英晶　振或陶　瓷晶　振		V_{DD}=4.5~5.5V	1	--	10	MHZ
		V_{DD}=2.7~4.5V	1	--	6	MHZ
		V_{DD}=2.0~2.7V	1	--	3	MHZ
外　部时　钟		V_{DD}=4.5~5.5V	1	--	10	MHZ
		V_{DD}=2.7~4.5V	1	--	6	MHZ
		V_{DD}=2.0~2.7V	1	--	3	MHZ
外　部RC振荡	--	V_{DD}=4.75~5.25V 偏差范围：10％	--	4	--	MHZ
内　部RC振荡		V_{DD}=4.75~5.25V		3.2		
振荡器				0.5		

表 13-5　　振荡稳定时间

（T_A=-40°C到+85°C，VDD=3.0V~5.5V）

振　荡　器	测 试 条 件	最小值	典型值	最大值	单位
石英晶振	f_{OSC}>1.0MHZ	--	--	20	ms
陶瓷振荡	当 V_{DD} 满足最小的起振电压时，晶振开始稳定。	--	--	10	ms
外部时钟	X_{IN} 输入的高低电平宽度	25	--	500	ns
晶　振稳定时间	外部复位操作所需稳定晶振的时间 t_{WAIT} (1)	--	2^{16}/ f_{OSC}	--	ms
等待时间	中断操作时，所需稳定晶振的时间 t_{WAIT} (2)	--	--	--	ms

1　f_{OSC} 是振荡器频率

2　当由于中断使系统推出空闲模式或冻结模式时，晶振的稳定时间是由 Basic timer 的控制寄存器决定的。

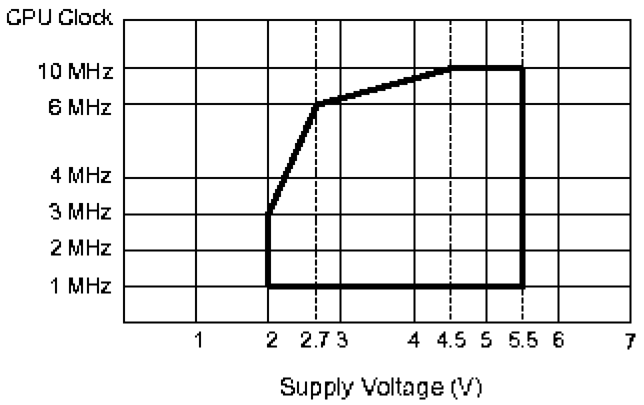


图 13-2　　操作电压范围

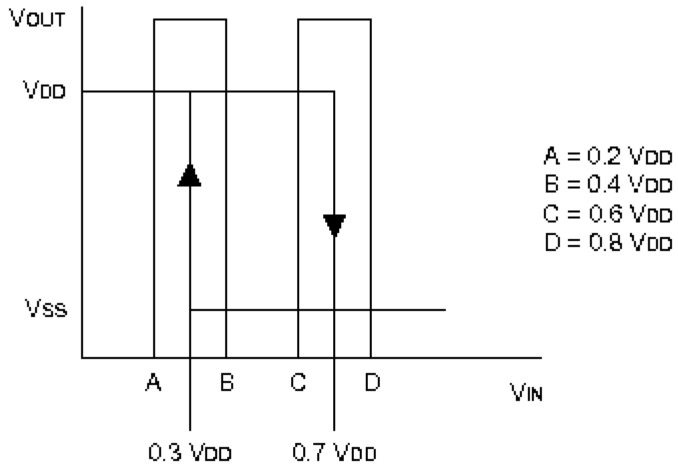


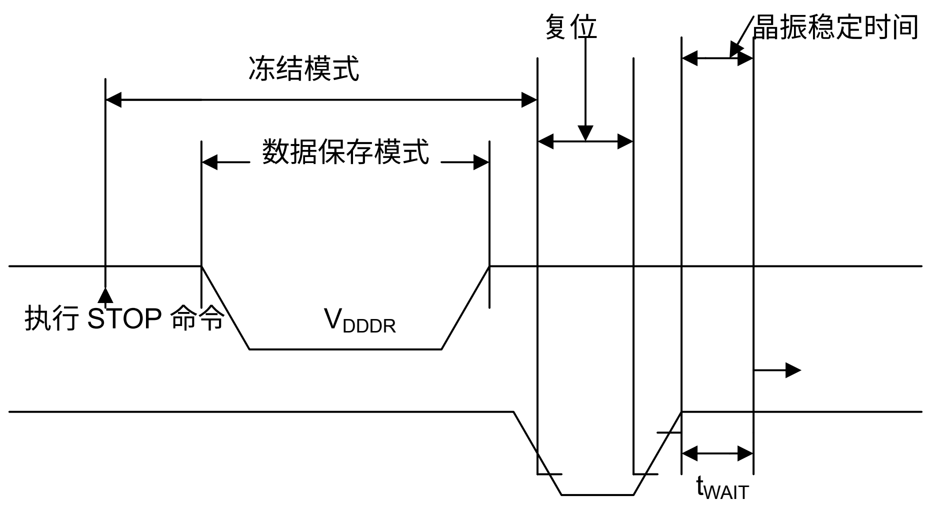
图 13-3　施密特触发器输入特性原理图

表 13-6　　在冻结状态（STOP）下保持数据所需供电电压

（T_A=-40°C到+85°C，VDD=2.0V~5.5V）

参　　数	标号	条　　件	最 小 值	典 型 值	最 大 值	单　　位
保持数据所需供电电　压	V_{DDDR}	冻 结 状 态 (STOP 模式)	2.0V	--	5.5	V
保持数据所需供电电　流	I_{DDR}	冻 结 状 态 (STOP 模式) V_{DDDR}=2.0V	0.1V	0.1	5	μA

注释：供电电流不包括内部上拉电阻的电流或者外部灌电流。



注释：t_{WAIT} 是稳定晶振所需要的时间，如 4096×16×1/f_{OSC}

图 13-4　外部复位使系统退出冻结状态示意图

表 13-7　　A/D 转换电气特性

（T_A=-40°C到+85°C，VDD=2.7V~5.5V，VSS=0V）

参　　数	符 号	测 试 条 件	最 小 值	典 型 值	最大值	单 位
精　　度	--	VDD=5.12V cpu 时钟=10M VSS=0V	--	--	±3	LSB
积分线性误差	ILE	”	--	--	±2	
微分线性误差	DLE	”	--	--	±1	
最高点偏移误差	EOT	”	--	±1	±3	
最低点偏移误差	EOB	”	--	±1	±2	
转换时间	t_{CON}	f_{OSC}=10MHZ	--	20	--	μS
模拟信号输入电压	V_{IAN}	--	VSS	--	VDD	V
模拟信号输入阻抗	R_{AN}	--	2	--	--	MΩ
模拟信号输入电流	I_{ADIN}	VDD=5V	--	--	10	μA
A/D 转换模块电流	I_{ADC}	VDD=5V	--	1	3	mA
		VDD=3V		0.5	1.5	
		VDD=5V 掉电模式下	--	100	500	nA

注释：1　转换时间是从启动转换到转换结束所需要的时间

2　I_{ADC}是在 A/D 转换过程中的电流

表 13-8　　LVR 电路特性

（T_A=-40°C到+85°C，VDD=2.0V~5.5V）

参　　数	符　号	条　　件	最　小值	典　型值	最　大值	单位
低电压复位	V_{LVR}	--	--	2.3 3.0 3.9		V
LVR 滞后电压	V_{HYS}		--	0.3	--	V
供电电压上升时间	t_R		10			μS
供电电压停止时间	t_{OFF}		0.5			S

注释：$2^{16}/f_x$(=6.55ms,在 10M 频率下)

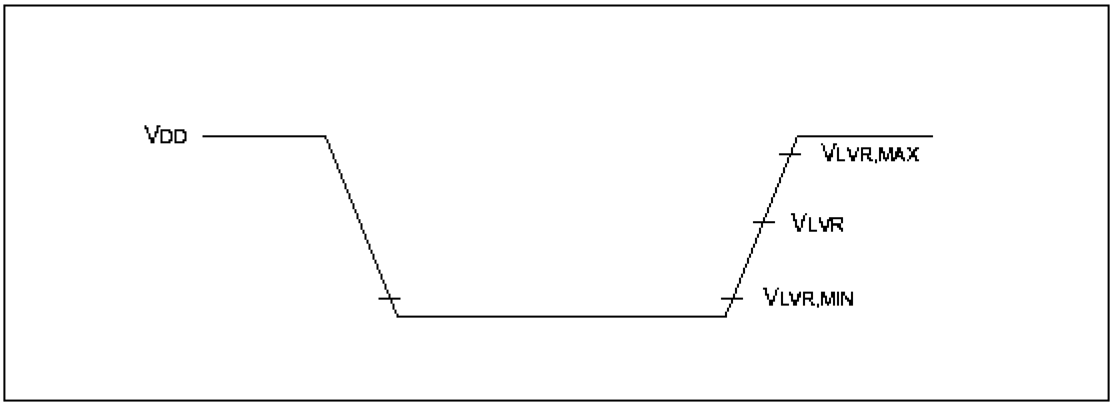


图 13-5　　LVR 复位时间

第14章　　　　机械尺寸

概述

实际应用中的S3C9454的封装形式有：20-Pin DIP封装（Samsung：20-DIP-300A），20-PIN SOP 封装（Samsung：20-SOP-375），20-Pin SSOP 封装（Samsung：20-SSOP-225），16-Pin DIP 封装（Samsung：16-DIP-300A），16-Pin SOP 封装（Samsung：16 –SOP-BD300-SG），16-Pin SSOP 封装（Samsung：16-SSOP-BD44），封装尺寸如图 14-1，14-2，14-3，14-4，14-5 和 14-6。

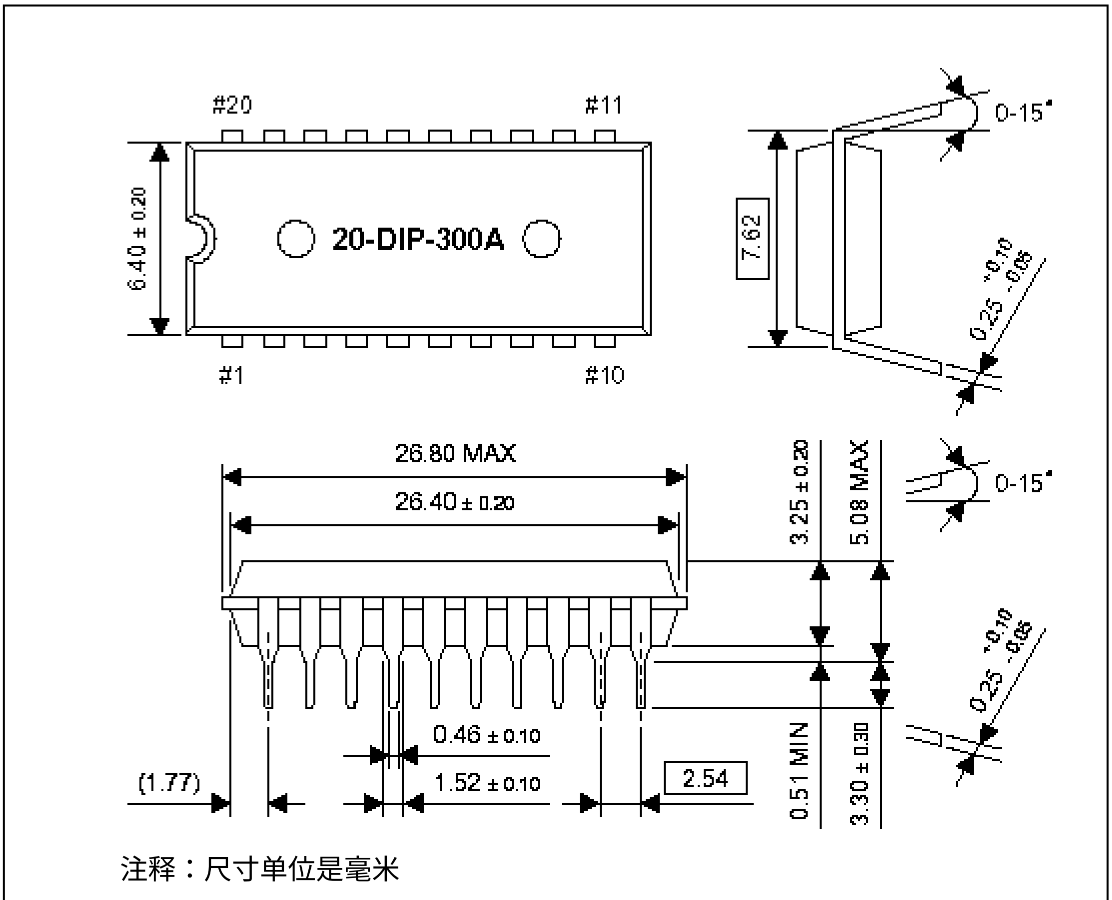
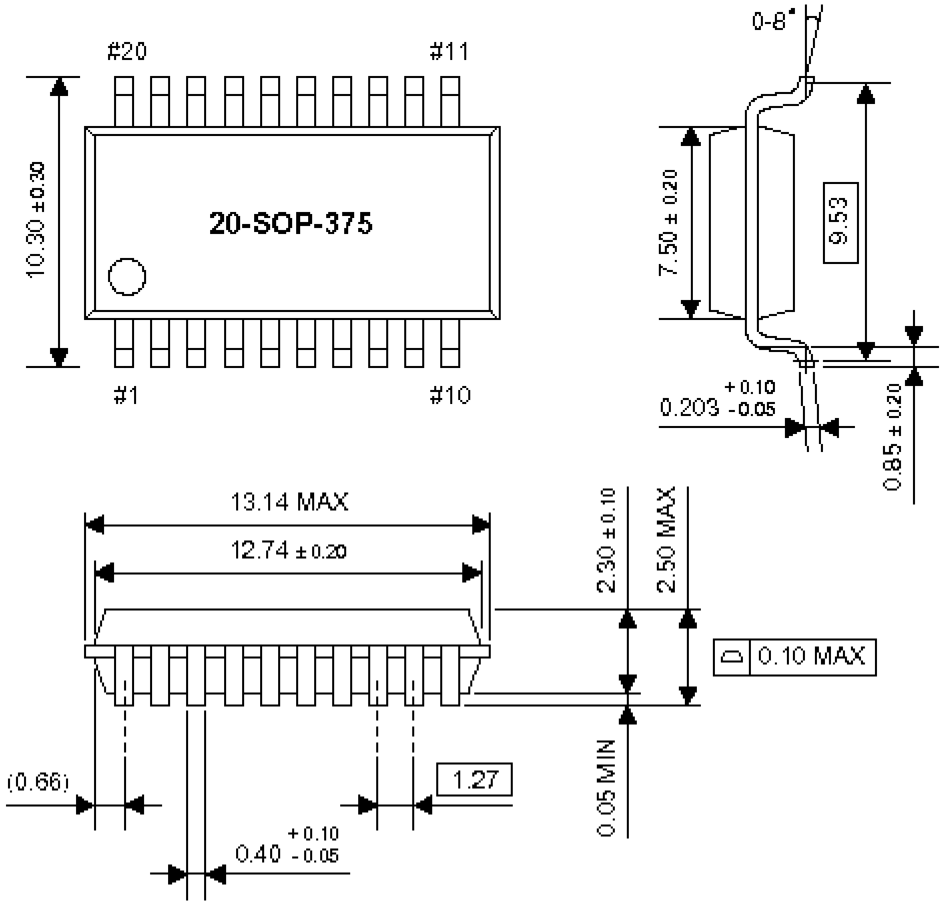
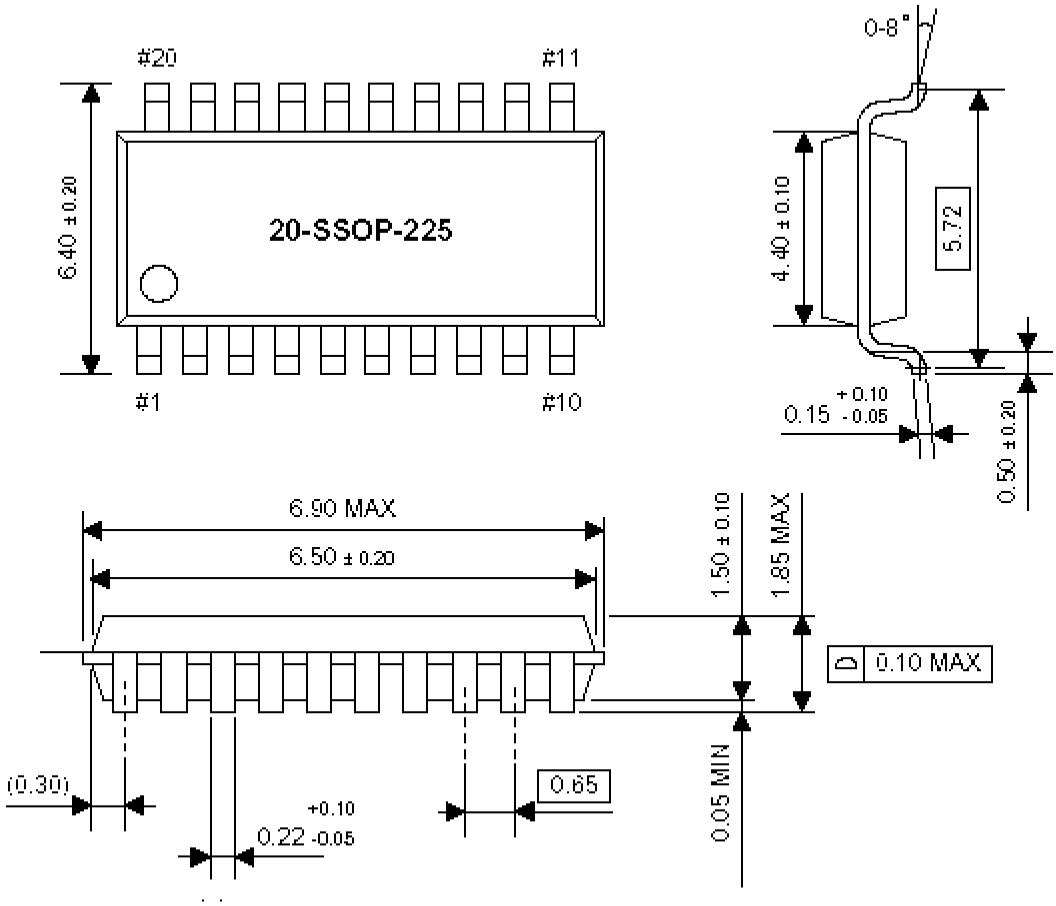


图 14-1　　20-DIP-300A 封装尺寸



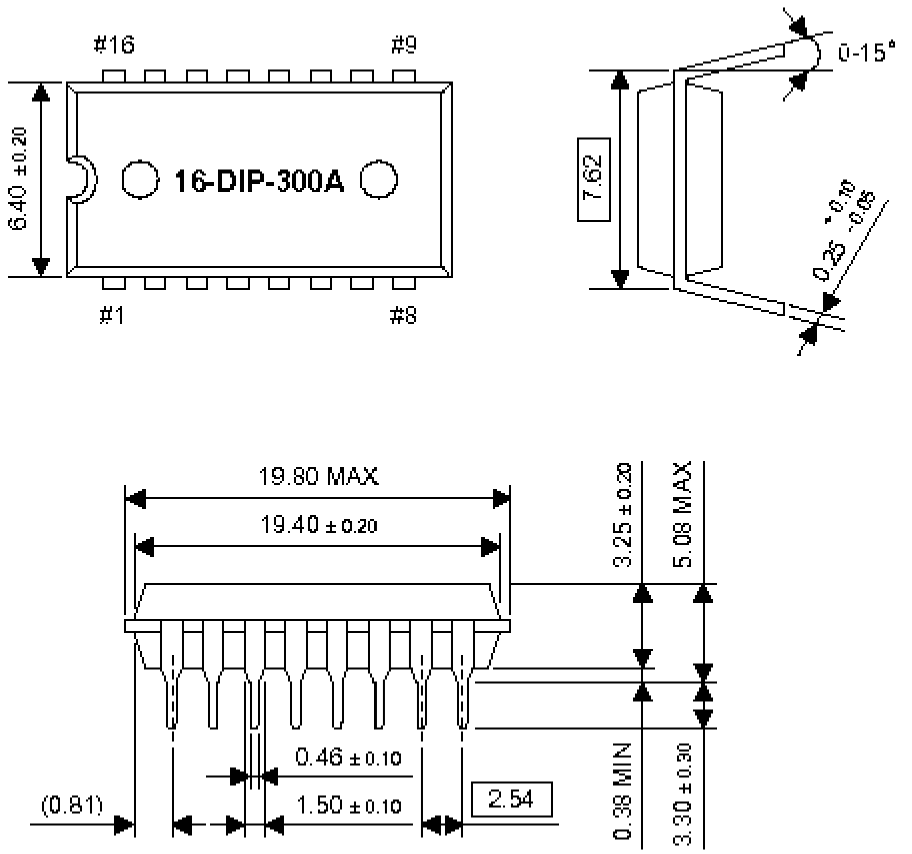
注释：尺寸单位是毫米（mm）

图 14-2　　　20-SOP-375 封装尺寸



注释：尺寸单位是毫米（mm）

图 14-3　　20-SSOP-225 封装尺寸



注释：尺寸单位是毫米（mm）

图 14-4　　16-DIP-300A 封装尺寸

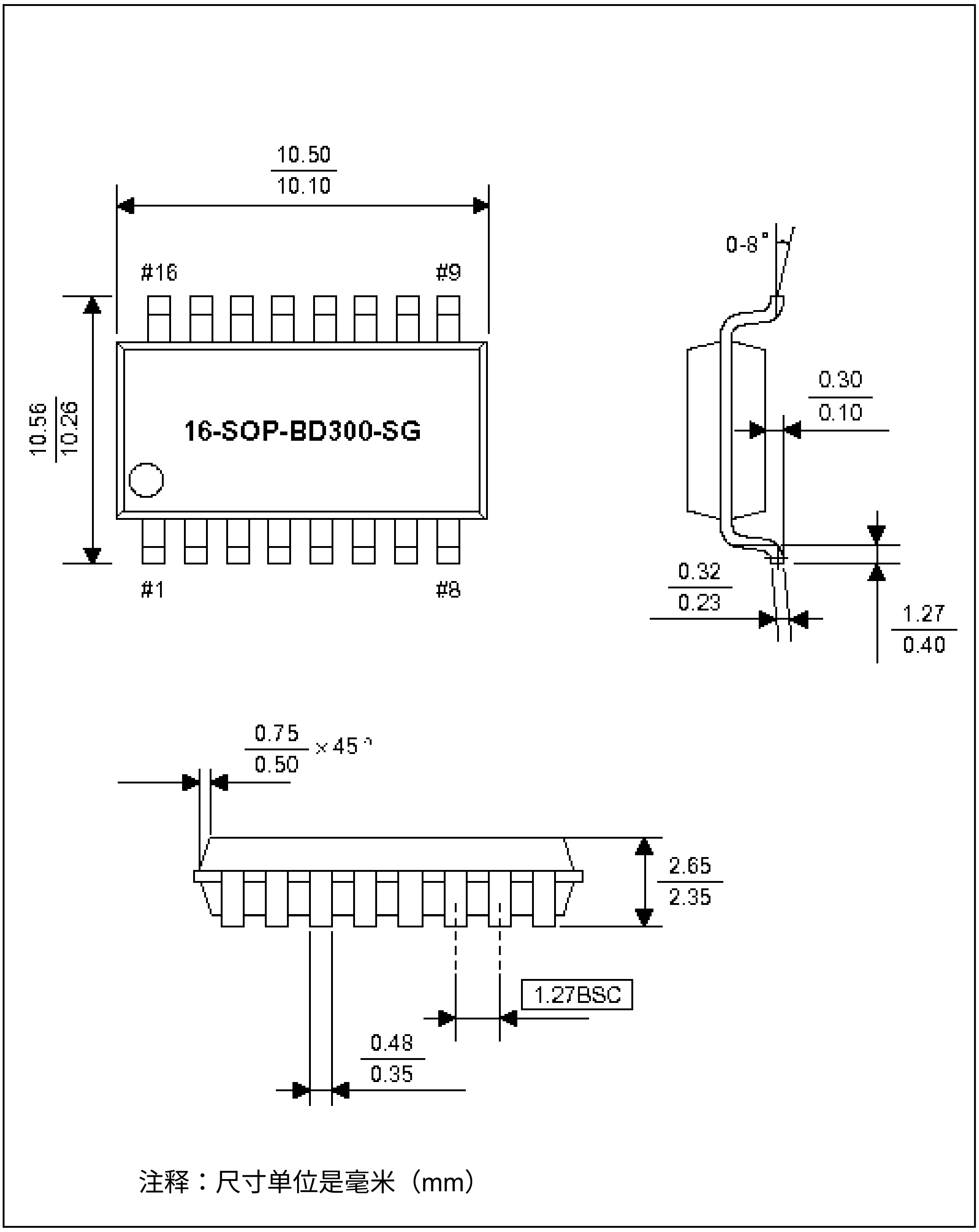


图 14-5　　16-SOP-BD300-SG 封装尺寸

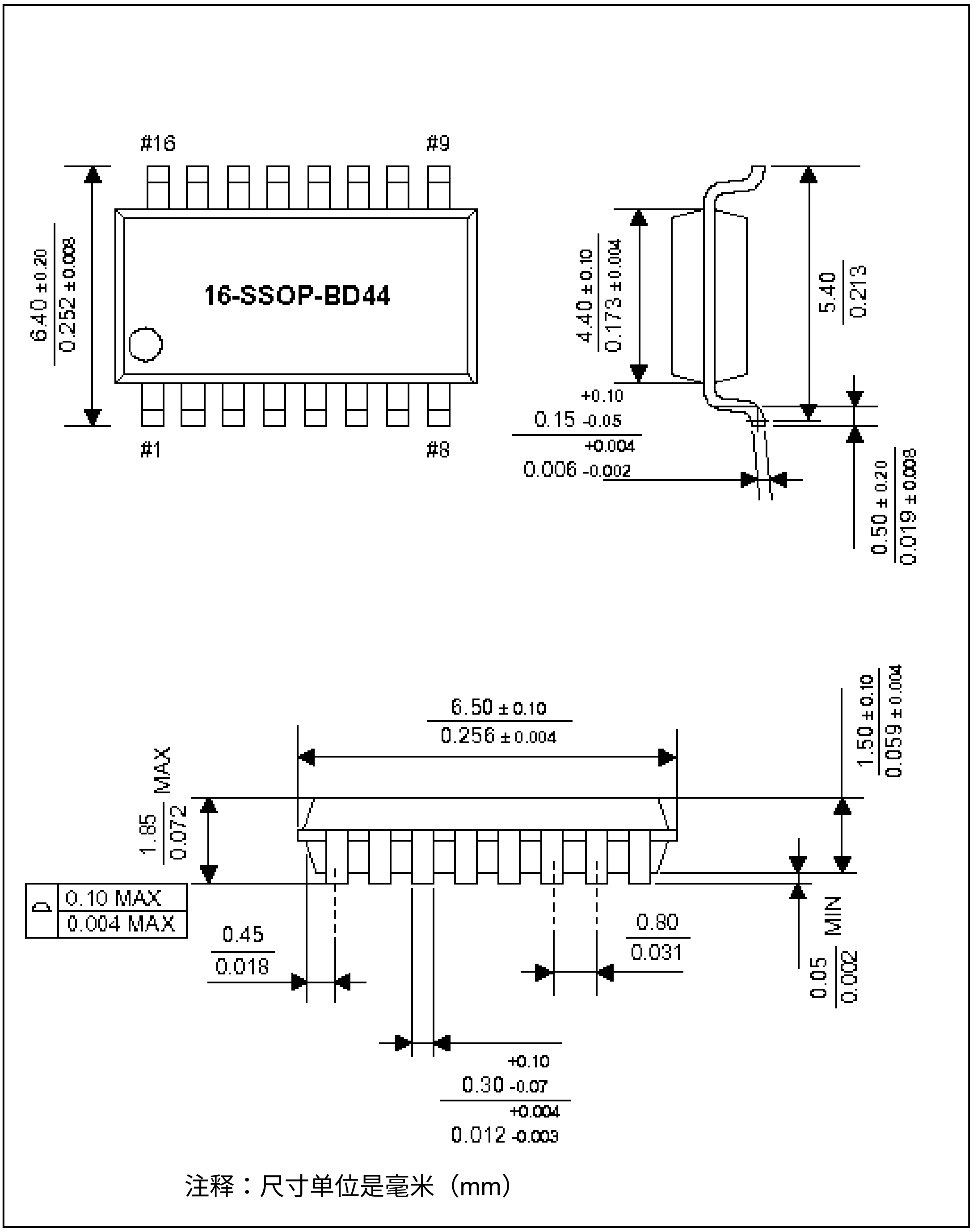


图 14-6　　16-SSOP-BD44 封装尺寸

第15章 S3F9454B MTP

概述

S3F9454 是一款可重复擦写（MTP）的单片 COMS 型微控制器。它内部用 FLASH ROM 代替了掩模 ROM。串行数据格式可以访问内部 FLASH ROM。

S3C9454 不管在直流电气特性还是在引脚特性上面与 S3C9454 完全兼容的。由于 S3F9454 编程的简易性，它是 S3C9454/F9454 芯片中的理想选择。

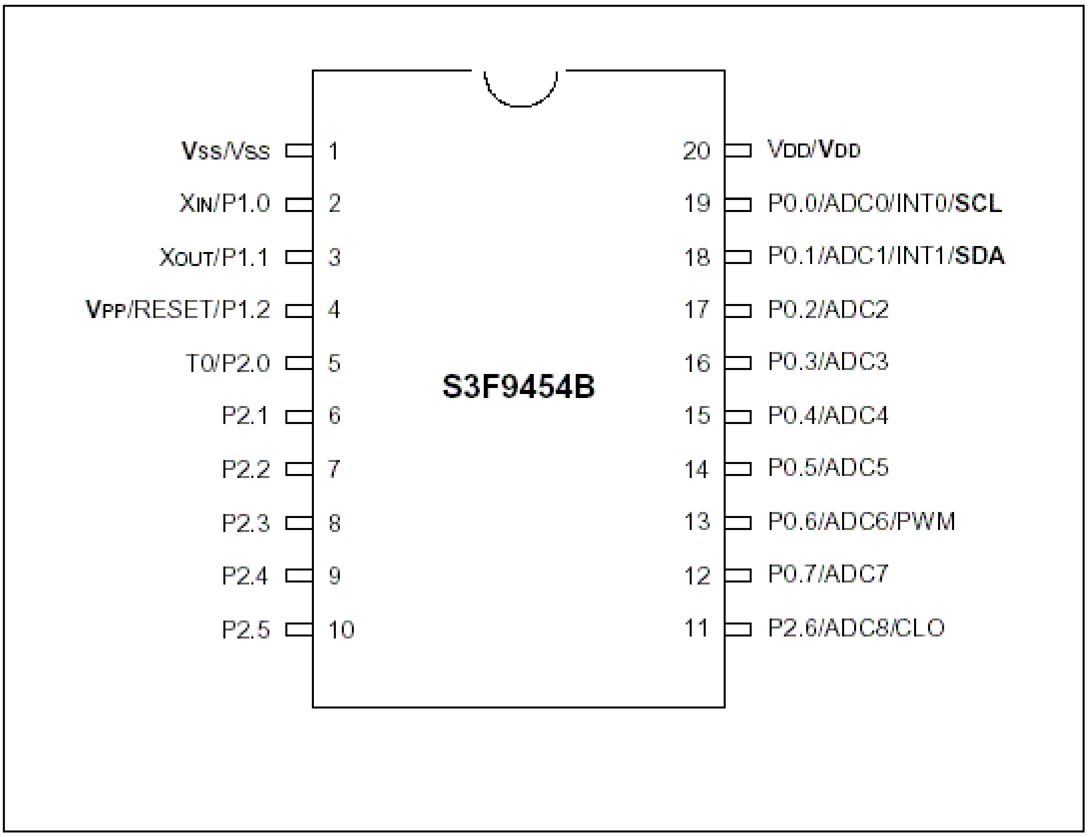


图 15-1 引脚分布图（20—Pin 封装）

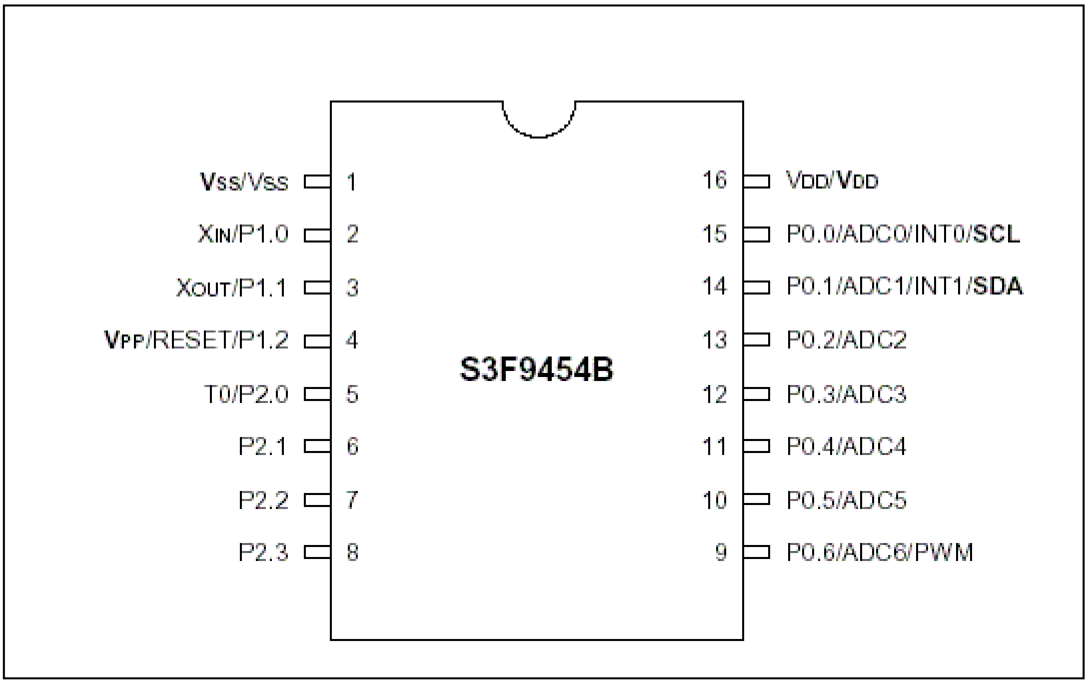


图 15-2　　引脚分布图（16-Pin 封装）

表 15-1　　读/写 FLASH ROM 引脚的描述

芯片的引脚名称	编程			
	引脚名称	引脚号	I/O	功能
P0.1	SDA	18(20-pin) 14(16-pin)	I/O	串行数据口（当读的时候,为输出；当写的时候，为输入），该口可设定为输入或推拉式输出
P0.0	SCL	19(20-pin) 15(16-pin)	I	串行时钟输入口（只有输入）
RESET, P1.2	V_{PP}	4	I	对 FLASH ROM 写时，提供的电源。当为 12.5V 时，MTP 处于写状态；当为 5V 时，MTP 处于读状态。
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20(20-PIN) 16(16-PIN) 1(20-PIN) 1(16-PIN)	I	提供芯片逻辑电压

表 15-2 S3F9454B 和 S3C9454B 的对比

特 性	S3F9454B	S3C9454
程 序 空 间	4K 字节的 FLASH ROM	4K 字节的掩模 ROM
工 作 电 压	2.0V~~5.5V	2.0V~~5.5V
MTP 编 程 模 式	VDD=5V，VPP=12.5V	
封 装 形 式	20DIP/20 SOP/20 SSOP/16 DIP/16 SOP/16 SSOP/8 DIP/8 SOP	
EPROM 可编程能力	用户可多次编程	在工厂编程

操作模式

当在 V_{PP} 引脚外加 12.5V 电压时，S3F9454B 进入编程模式。下表给出了，根据 V_{PP} 引脚的电压，S3F9454 所进入的操作模式（读、写、或读保护）。

表 15-3 操作模式的选择标准

V_{DD}	V_{PP}	REG/MEM	Address（A15-A0）	R/W	模 式
5V	5V	0	0000H	1	读 FLASH ROM
	12.5V	0	0000H	0	编程 FLASH ROM
	12.5V	0	0000H	1	校验 FLASH ROM
	12.5V	1	0000H	0	FLASH ROM 读保护

注释：‘0’表示低电平，‘1’表示高电平。

第16章　　开发工具

概述

三星为用户提供了强大易用的开发系统平台。开发系统平台由主机，调试工具和支持软件组成。对于主机，任何安装 Win98/2000/XP 操作系统的标准计算机都可以运用此开发系统。其中一种形式的调试工具包括软件和硬件，它内部有针对 S3C7，S3C9，S3C8 系列微控制器的仿真器—OPENice-i500。OPENice-i500 是由第三方(AIJI System)提供的开发工具。AIJI System 提供的 OPENice-i500 软件，包括调试器、汇编器以及编程设置项。

SASM 汇编器

SASM 是一款针对 S3C9 系列微控制器可重新定位的汇编器。SASM 可以编辑用汇编语句编写的源程序并汇编为相应的源代码、目标文件和注释语句。SASM 支持宏汇编和条件汇编。她运行于 MS-DOS 或 Win98/2000/XP 环境，同时，她只生成目标代码，所以用户需要连接目标文件，目标文件可以包含有其它目标文件并装入内存。SASM 汇编器需要源文件和特定芯片的寄存器文件（device_name.reg）

SAMA 汇编器

SAMA 是一款通用的汇编器，能生成标准的 16 进制格式的目标代码。汇编过的程序代码包括目标代码，这些目标代码是用于 ROM 的数据和必需的内部电路模拟器的程序控制数据。对于汇编程序，SAMA 需要源文件和特定芯片的定义文件（DEF 文件）。DEF 文件用于描述器件的特定信息。

OPENice—SLD

OPENice—SLD 主操作界面是基于 OPENice-i500/i300 的多窗口仿真调试软件。OPENice-SLD 支持下拉式、弹出式菜单，支持鼠标，热键和连接式帮助文档。同时，它是一款简单易用的多窗口用户界面，支持窗口的大小改变、移动、滚动。

HEX2ROM

HEX2ROM 软件包把由汇编器生成的 HEX 文件生成为 ROM 代码。对于带掩模 ROM 的微控制器，这些代码是必须的。HEX2ROM 软件包在生成 ROM 代码时自动把目标板 ROM 区没有用到的 ROM 填充为‘FFH’。

目标板

对于 S3C9 系列的微控制器都需要用到目标板。每种特定的目标板都配有目标系统电缆和适配器。

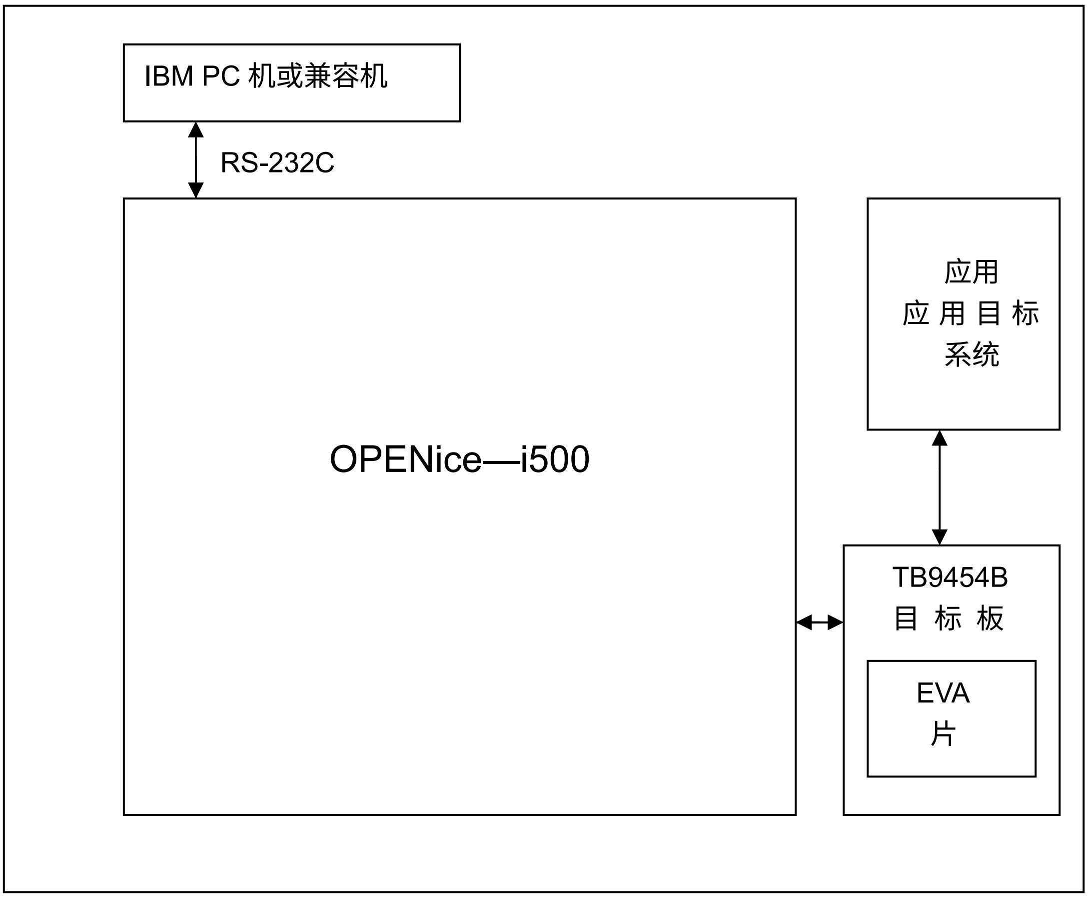


图 16-1　OPENice—i500 产品结构

TB9454B 目标板

TB9454B 目标板用于 S3C9454B/F9454B 的开发，OPENICE-I500 支持此目标板。

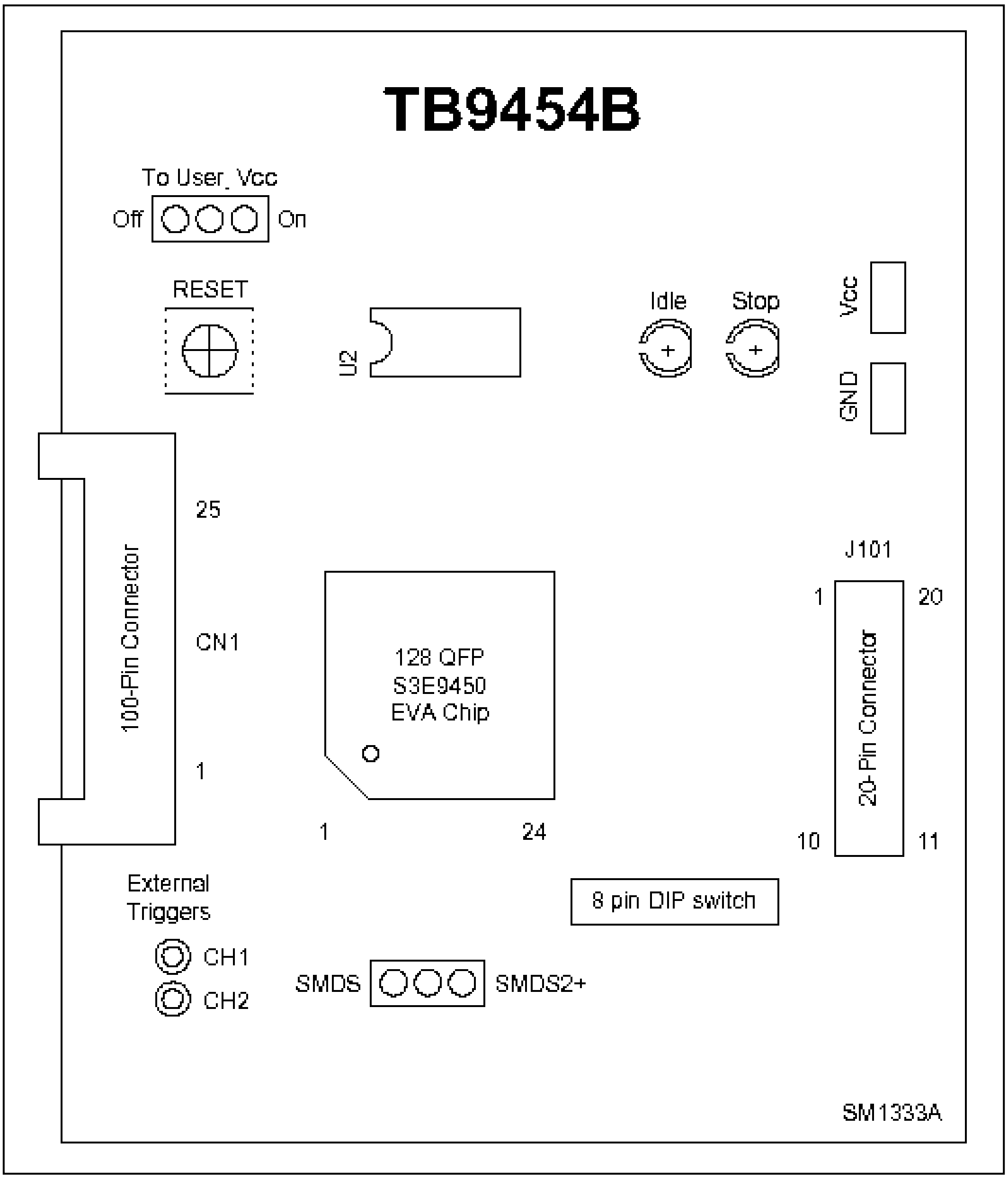
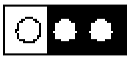
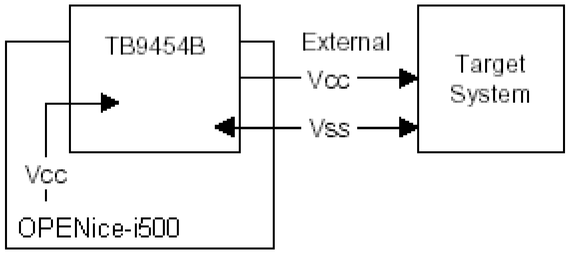
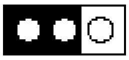
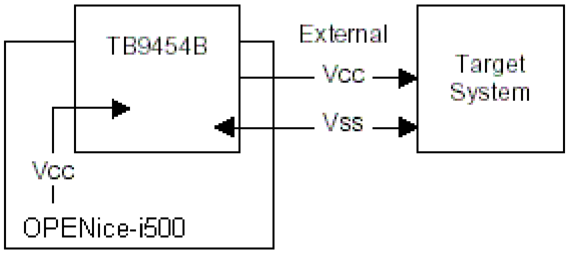
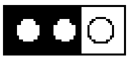


图 16-2　　TB9454B 目标板结构

表 16-1　　TB9454B 的电源选择设置

到用户板的电源设置	操　作　模　式	备　注
用户电源开 off  on		OPENice-i500 主板电源供给目标板和目标系统
用户电源关 off  on		OPENice-i500 主板电源供给目标板，目标系统得电源需要自己提供

注意：在“ 用户板电源设置 ”栏中，下面的符号表示电源短路配置



SMDS2+选择（SAM8）

为了把 SMDS2+用到的数据写到程序存贮区，在目标板上应该如下图配置，否则，程序存贮区的写功能不能应用。

表 16-2　　SMDS2+工具选择设置

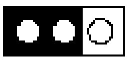
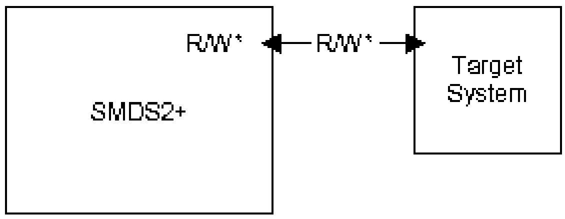
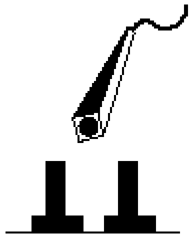
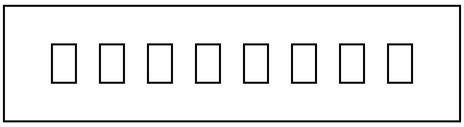
“SW1”设置	操作模式
SMDS  SMDS2+	

表 16-3　　用单个插针作为外部触发源的输入端

目标板部分	备注
外 部 触 发 ◎　Ch1 ◎　Ch2	 从应用系统输入的外部触发信号连接线 用户可以通过其中的任一插针把外部触发信号引入，作为 OPENice-i500 的断点和跟踪功能

ON



OFF

开	低电平
关	高电平

注释：关于 EVA，smart option 是由 DIP 开关决定的，
并不是由软件决定的

图 16-3　　Smart Option 的 DIP 开关

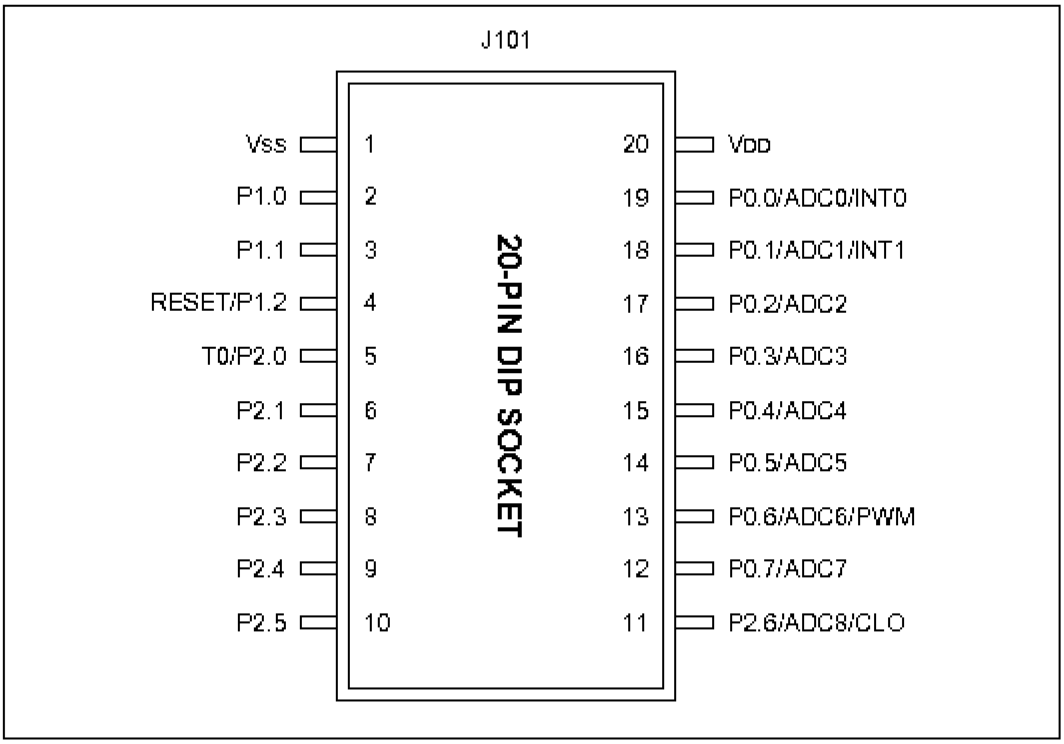


图 16-4　　20-Pin TB9454B 插座

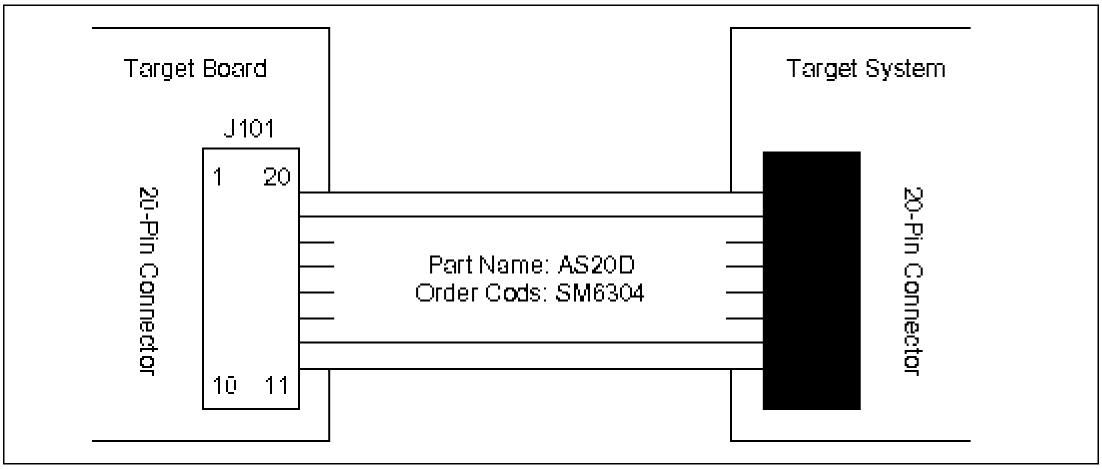


图 16-5　20-DIP 封装 S3C944B/F9454B 适配器